

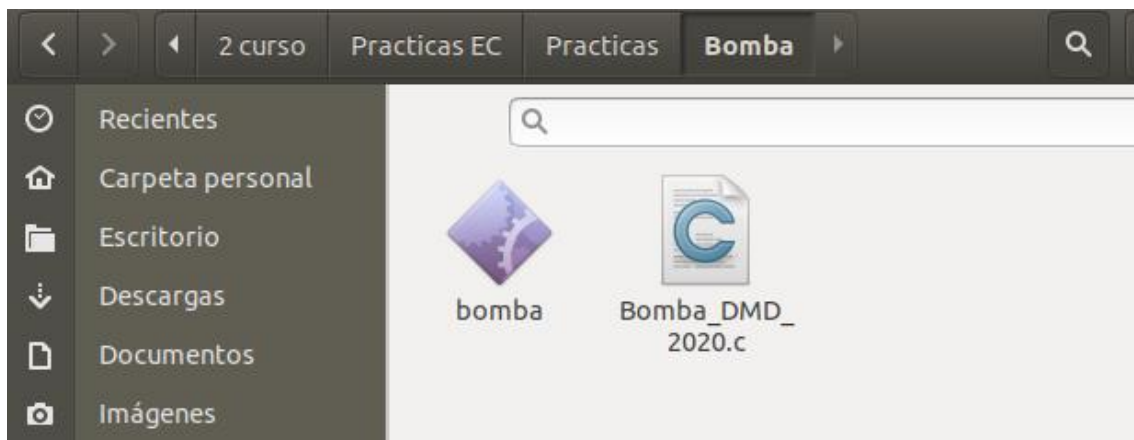
Bomba Digital

David Martínez Díaz

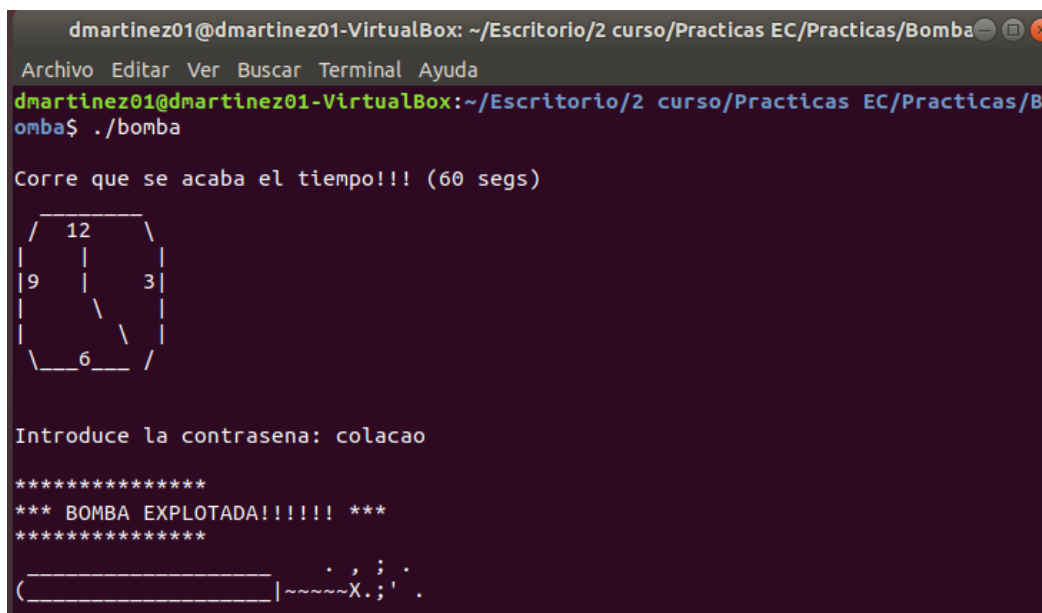
1.- En primer lugar, debemos obtener la bomba correspondiente, en mi caso, utilizare la bomba que yo he creado ("Bomba_DMD_2020.c"), al cual debemos aplicarle el siguiente comando:

```
*gcc -Og Bomba_DMD_2020.c -o bomba -no-pie -fno-guess-branch-probability*
```

Pudiendo utilizar distintos tipos de optimizaciones, entre las que podemos destacar desde la `-Og` hasta el `-O2`. Creándose su correspondiente ejecutable.



2. En segundo lugar es ejecutarlo, donde vemos que tenemos que averiguar su contraseña y pin correspondiente.



3. Para ello debemos introducirnos en dicho ejecutable a través del debug e ir mirando los registros para comprobar y obtener las contraseñas.

→ gdb bomba

→ layout regs

Una vez estamos debugeando, hay que buscar el lugar exacto donde se comparan las contraseñas para poder continuar con el ejercicio, donde si nos damos cuenta, esta se realiza en el string compare, en la línea (*main+130).

```
dmartinez01@dmartinez01-VirtualBox: ~/Escritorio/2 curso/Practicas EC/Practicas/Bomba
Archivo Editar Ver Buscar Terminal Ayuda
Register group: general
rax      0x64    100      rbx      0xff     255
rcx      0x63    99       rdx      0x9       9
rsi      0x7fffffffddc0 140737488346560 rdi      0x7fffffffddc0 140737488346560
rbp      0x4008b0 0x4008b0 <_libc_csu_init> rsp      0x7fffffffdd90 0x7fffffffdd90
r8       0x602672 6301298 r9       0x7ffff7fe2500 140737354016000
r10      0x602010 6299664 r11      0x246     582

0x4007cc <main+113> lea    0x30(%rsp),%rdi
0x4007d1 <main+118> mov    $0x9,%edx
0x4007d6 <main+123> lea    0x20088b(%rip),%rsi      # 0x601068 <password>
0x4007dd <main+130> callq 0x4005d0 <strncmp@plt>
0x4007e2 <main+135> test   %eax,%eax
0x4007e4 <main+137> je     0x4007ff <main+164>

native process 2429 In: main
(gdb) b *main+123
Punto de interrupción 1 at 0x4007d6
(gdb) run
Starting program: /home/dmartinez01/Escritorio/2 curso/Practicas EC/Practicas/Bomba/bomba
Breakpoint 1, 0x000000004007d6 in main ()
(gdb)
```

Por tanto, para sacar su contraseña debo ver los valores que hay en el registro %rsi, viendo así su contraseña correspondiente:

```
Register group: general
rax      0x5      5      rbx      0xff     255
rdx      0x0      0      rsi      0x601068 6295656
rbp      0x4008b0 0x4008b0 <_libc_csu_init> rsp      0x7fffffffdd90 0x7fffffffdd90
r9       0x7ffff7fe2500 140737354016000 r10      0x3       3
r12      0x400640 4195904 r13      0x7ffff7fdf10 140737488346896
r15      0x0      0      rip      0x4007e4 0x4007e4 <main+137>
cs       0x33     51      ss       0x2b     43
es       0x0      0      fs       0x0      0

0x4007cc <main+113> lea    0x30(%rsp),%rdi
0x4007d1 <main+118> mov    $0x9,%edx
0x4007d6 <main+123> lea    0x20088b(%rip),%rsi      # 0x601068 <password>
0x4007dd <main+130> callq 0x4005d0 <strncmp@plt>
0x4007e2 <main+135> test   %eax,%eax
0x4007e4 <main+137> je     0x4007ff <main+164>
0x4007e6 <main+139> callq 0x400727 <boom>
0x4007eb <main+144> movslq %eax,%rcx
0x4007ee <main+147> movzbl 0x30(%rsp,%rcx,1),%ebx
0x4007f3 <main+152> lea    0x5(%rbx),%edx
0x4007f6 <main+155> mov    %dl,0x30(%rsp,%rcx,1)
0x4007fa <main+159> add    $0x1,%eax
0x4007fd <main+162> jmp    0x4007c7 <main+108>
0x4007ff <main+164> lea    0x20(%rsp),%rdi

native process 2941 In: main
Breakpoint 1, 0x000000004007d6 in main ()
(gdb) ni
0x000000004007dd in main ()
(gdb) ni
0x000000004007e2 in main ()
(gdb) ni
0x000000004007e4 in main ()
(gdb) x/s $rsi
0x601068 <password>: "hmnhpjsx\017"
```

Sin embargo, nos encontramos con el primer problema, si introducimos la contraseña nos explotara la bomba:

```
dmartinez01@dmartinez01-VirtualBox:~/Escritorio/2 curso/Practicas EC/Practicas/Bomba$ ./bomba

Corre que se acaba el tiempo!!! (60 segs)

  ____  _
 /  __ \| | | |
| |  | | | | | |
| |  | | | | | |
| |  | | | | | |
 \  __ \| | | |
  ____  _ | | | |

Introduce la contraseña: hmnhpjsx

*****
*** BOMBA EXPLOTADA!!!!!! ***
*****

(_____|~~~~X.;' .

dmartinez01@dmartinez01-VirtualBox:~/Escritorio/2 curso/Practicas EC/Practicas/Bomba$
```

Esto quiere decir que la contraseña esta encriptada, y no vamos a poder poner directamente el resultado de %rsi, para ello podemos comprobar con nuestra contraseña anterior y mirar en el registro %rdi para ver dicha comparación y calcular esa razón correspondiente para saber cuánto hay que sumarlo o restarle a esta.

Por ello si introducimos la contraseña “hola” y nos introducimos en el registro %rdi obtenemos lo siguiente:

```
(gdb) x/s $rdi
0x7fffffffddc0: "mtqf\017", '\005' <repetidos 11 veces>, "\016\005\005\005\005\005\005\005\005e[\342\374\004\204\005\005M\343\004\004\004\204\005\005\004\272\365\005\005\005\005\005\006\005\005\005\005\005\005\002\rE\005\005\005\005e@\343\374\004\204\005\005\005\005\005\005\005\005\005\265\rE\005\005\005\005\005E\005\005\005\005\005\025\344\004\004\377\177"
(gdb)
```

Donde nuestra nueva contraseña encriptada seria “mtqf”, para saber la razón de encriptación simplemente basta saber que valor llega desde la primera letra de nuestra contraseña hasta la primera letra de la contraseña encriptada:

“hola” ---> “mtqf”;

“h” ---> “m”

Para sacarlo nos vamos a la tabla ASCII y vemos sus correspondientes valores:

“h” = 104 // “m” = 109

Con esto llegamos a la conclusión de que la razón es 5, y simplemente para saber su contraseña hay que restarle 5 a cada letra.

Consiguiendo así la contraseña encriptada:

“hmnhpjsx” ---> “chickens”;

Vamos a comprobarlo:

```
dmartinez01@dmartinez01-VirtualBox:~/Escritorio/2 curso/Practicass EC/Practicass/B
omba$ ./bomba

Corre que se acaba el tiempo!!! (60 segs)

  12
 /  \
|    |
| 9  | 3 |
|    |
|  \  /
| 6  |
 \  /

Introduce la contrasena: chickens
Introduce el pin: █
```

Vemos que la contraseña era correcta pero se nos presenta otra problema, ahora nos pide un pin, veamos entonces de nuevo en el gdb, donde se encuentra este.

Como podemos ver se realiza otro string compare el cual vamos a analizar para saber si ahí se realiza la comparación entre pines. Analizando las sentencias llegamos hasta el main+272, donde se realiza una comparación entre los registros con la orden “jne”:

```
Register group: general
rax      0x1      1
rdx      0x7ffff7dcf8d0  140737351842000
rbp      0x4008b0  0x4008b0 <__libc_csu_init>
r9       0x0      0
r12      0x400640  4195904
r15      0x0      0
cs       0x33     51
es       0x0      0
rbx      0x1      1
rsi      0x1      1
rsp      0x7fffffffdd90  0x7fffffffdd90
r10      0x7ffff7b80c40  140737349422144
r13      0x7fffffffdf10  140737488346896
rip      0x40086b  0x40086b <main+272>
ss       0x2b     43
fs       0x0      0

0x400853 <main+248>    jne    0x400866 <main+267>
0x400855 <main+250>    lea    0x2a6(%rip),%rdi    # 0x400b02
0x40085c <main+257>    mov    $0x0,%eax
0x400861 <main+262>    callq 0x400620 <__isoc99_scanf@plt>
0x400866 <main+267>    cmp    $0x1,%ebx
0x400869 <main+270>    jne    0x400823 <main+200>
3+> 0x40086b <main+272>    mov    0x2007ef(%rip),%eax    # 0x601060 <passcode>
0x400871 <main+278>    cmp    %eax,0xc(%rsp)
0x400875 <main+282>    je     0x40087c <main+289>
0x400877 <main+284>    callq 0x400727 <boom>
0x40087c <main+289>    lea    0x10(%rsp),%rdi
0x400881 <main+294>    mov    $0x0,%esi
0x400886 <main+299>    callq 0x4005f0 <gettimeofday@plt>
0x40088b <main+304>    mov    0x10(%rsp),%rax
0x400890 <main+309>    sub    0x20(%rsp),%rax
```

Donde si miramos bien las sentencias y hacemos un par de next instructions vemos como sacan un dato de la pila en la dirección de (%rip+0x2007ef), y lo mete en el registro %eax, si lo mostramos vemos que:

```
(gdb) print $eax
$1 = 2544
(gdb) █
```

Vamos a comprobar si este pin es el correcto:

```
Corre que se acaba el tiempo!!! (60 segs)

  12
  |
9 | 3
  |
  \
  6

Introduce la contraseña: chickens
Introduce el pin: 2544

---- NOS SALVAMOS!!!! ----

  12
  |
9 | 3
  |
  \
  6

dmartinez01@dmartinez01-VirtualBox:~/Escritorio/2 curso/Practicas EC/Practicas/B
omba$
```

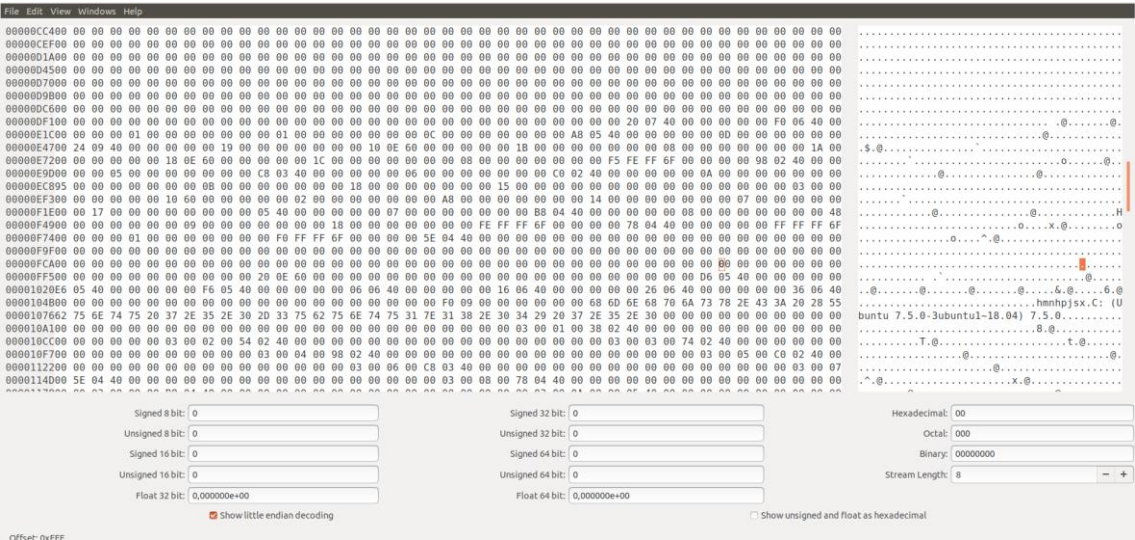
Como hemos podido comprobar el pin que habíamos obtenido era correcto y no se encontraba encriptado por lo que hemos podido desactivar la bomba. Por lo que esta sería la manera de desactivar mi bomba.

- **Modificar la contraseña de la bomba:**

Para modificar la contraseña vamos a utilizar el comando ghex que permite ver el código de manera hexadecimal:

“ghex Bomba_DMD_2020”

Donde nos aparecerá lo siguiente:



Una vez estamos dentro, tenemos que buscar dicha contraseña, que en mi caso se encuentra encriptada:

```
.....@.....  
..@.....@.....@.....@.....&@.....6.@  
.....hmnhpjxs.C: (U  
buntu 7.5.0-3ubuntu1~18.04) 7.5.0.....  
.....8.@.....
```

Una vez que la hemos sacado vemos que su Offset es 0x106F, simplemente si la modificamos debería cambiar (teniendo en cuenta la razón de encriptación).

Por ejemplo, si yo cambio la contraseña de esta manera:

“chickens” → “hmnhpjxs”.

Si yo cambiase la “x” del final por una “p”, aplicándole la razón de 5 en este caso:

Se quedaría el código de tal manera que:

“chickenk” → “hmnhpjsp”.

```
dmartinez01@dmartinez01-VirtualBox:~/Escritorio/Practica 4$ ghex bomba  
dmartinez01@dmartinez01-VirtualBox:~/Escritorio/Practica 4$ ./bomba  
Corre que se acaba el tiempo!!! (60 segs)  
  
Introduce la contraseña: chickenk  
Introduce el pin:
```

Vemos que si me acepta la contraseña y la hemos podido modificar con éxito.

Para modificar el pin, vamos a emplear otra vez el comando ghex,

Signed 8 bit:	<input type="text" value="-16"/>	Signed 32 bit:	<input type="text" value="0"/>
Unsigned 8 bit:	<input type="text" value="240"/>	Unsigned 32 bit:	<input type="text" value="0"/>
Signed 16 bit:	<input type="text" value="2544"/>	Signed 64 bit:	<input type="text" value="0"/>
Unsigned 16 bit:	<input type="text" value="2544"/>	Unsigned 64 bit:	<input type="text" value="0"/>
Float 32 bit:	<input type="text" value="3,564903e-42"/>	Float 64 bit:	<input type="text" value="0"/>

☒ Show little endian decoding

Signed 8 bit: -12

Unsigned 8 bit: 244

Signed 16 bit: 2548

Unsigned 16 bit: 2548

Float 32 bit: 3,570508e-42

☒ Show little endian decoding

Por ultimo vamos a comprobar si funciona:



```
dmartinez01@dmartinez01-VirtualBox: ~/Escritorio/Practica 4
Archivo Editar Ver Buscar Terminal Ayuda
Corre que se acaba el tiempo!!! (60 segs)
  12
 /  \
|9   |3|
|    | |
|    | |
 \   /
  6

Introduce la contraseña: chickenk
Introduce el pin: 2548
---- NOS SALVAMOS!!!! ----
  _ _ _ _ _
 / _ _ _ _ \
| _ _ _ _ |
| _ _ _ _ |
| _ _ _ _ |
 \ _ _ _ _ /
  _ _ _ _ _

dmartinez01@dmartinez01-VirtualBox:~/Escritorio/Practica 4$
```

Y vemos que se ha modificado con éxito.