

ESTRUCTURA DE COMPUTADORES - 2º GRADO INGENIERÍA INFORMÁTICA PRÁCTICA 4.- BOMBA DIGITAL – DESENSAMBLADORES

Mario Antonio López Ruiz - 45109755Q

BOMBA DE JUAN EMILIO GARCIA MARTÍNEZ. COMO DESACTIVARLA

1.INICIO

Primero tengo que darle permisos al archivo para poder ejecutarlo → "chmod a+x bomba".

Para comenzar abro dos terminales, en uno ejecuto "objdump -d bomba" para tener la información en ensamblador a mano, y en el otro abro el archivo con "gdb bomba" (es el depurador que voy a utilizar)

```
Terminal
tehrribbon@tehrribbon-SATELLITE-L50-B: ~/Escritorio/INFORMATICA/curso_recu/mestre/EC/PRACTICAS/señon4/bombas_desactivadas/Juane_garcía$ objdump -d bomba.o
formato del fichero elf32-l386

Desensamblado de la sección .init:
08048420 <.init>:
08048420:      53                push    %ebx
08048421:      83 ec 08          sub     $0x8,%esp
08048424:      e8 17 01 00 00   call    8048540 <__x86.get_pc_thunk
                                BOMBA DE JUANE GARCÍA: COMO DESA
                                ESTRUCTURA DE COMPUTADORES -
                                2º GRADO INGENIERÍA INFORMÁTICA
                                PRÁCTICA 4.- BOMBA DIGITAL - DESEN
                                Modulo 4: Bombas de depuración 45109755Q
                                Por favor, no copiar ni reproducir en un
                                medio electrónico sin el consentimiento
                                de los autores.
                                gdb bombo
08048429:      81 c3 d7 1b 00 00 add     $0xb7d,%ebx
0804842f:      8b 83 fc ff ff ff mov     -0x4(%ebx),%eax
08048435:      85 c0             test    %eax,%eax
08048437:      74 05            je      804843e <.init+0x1e>
08048439:      e8 c2 00 00 00   call    8048500 <__isoc99_scanf@plt
+0x10>
0804843e:      83 c4 08          jmp     8048440
08048441:      5b              pop     %ebx
08048442:      c3              ret

Desensamblado de la sección .plt:
08048450 <printf@plt+0x10>:
08048450:      ff 35 04 a0 04 08 pushl   0x804a004
08048456:      ff 25 08 a0 04 08 jmp     *0x804a008
0804845c:      00 00           add     %al,(%eax)
...
08048460 <printf@plt>:
08048460:      ff 25 0c a0 04 08 jmp     *0x804a00c
08048466:      68 00 00 00 00 push    $0x0
0804846b:      e9 e0 ff ff ff   jmp     8048450 <.init+0x30>

08048470 <fgets@plt>:
08048470:      ff 25 10 a0 04 08 jmp     *0x804a010
08048476:      68 08 00 00 00 push    $0x8
0804847b:      e9 d0 ff ff ff   jmp     8048450 <.init+0x30>
```

En este punto, puedo empezar a ver que ocurre en el programa.

2.CONTRASEÑA

Pongo un primer punto de ruptura en el main con "break main" , comienzo con "run" voy avanzando con "nexti" hasta que me pide la contraseña. Introduzco un valor cualquiera y veo en que punto se activa.

En 0x08048772 es la ultima línea que se ejecuta, voy al terminal con objdump y observo que ocurre ahí:

```

Terminal
tehribbon@tehribbon-SATELLITE-L50-B: ~/Escritorio/INFORMATICA/curso_recu/

8048705: 8d 4c 24 04      lea    0x4(%esp),%ecx
8048709: 83 e4 f0         and    $0xfffff0,%esp
804870c: ff 71 fc         pushl  ~0x4(%ecx)
804870f: 55              push   %ebp
8048710: 89 e5           mov    %esp,%ebp
8048712: 51              push   %ecx
8048713: 81 ec 84 00 00 00 sub    $0x84,%esp
8048719: 65 a1 14 00 00 00 mov    %gs:0x14,%eax
804871f: 89 45 f4         mov    %eax,-0xc(%ebp)
8048722: 31 c0           xor     %eax,%eax
8048724: 83 ec 08         sub    $0x8,%esp
8048727: 6a 00           push   $0x0
8048729: 8d 45 80         lea    -0x80(%ebp),%eax
804872c: 50              push   %eax
804872d: e8 4e fd ff ff   call   8048480 <gettimeofday@plt>
8048732: 83 c4 10         add    $0x10,%esp
8048735: 83 ec 0c         sub    $0xc,%esp
8048738: 68 f4 88 04 08   push   $0x80488f4
804873d: e8 1e fd ff ff   call   8048460 <printf@plt>
8048742: 83 c4 10         add    $0x10,%esp
8048745: a1 60 a0 04 08   mov    0x804a060,%eax
804874a: 83 ec 04         sub    $0x4,%esp
804874d: 50              push   %eax
804874e: 6a 64           push   $0x64
8048750: 8d 45 90         lea    -0x70(%ebp),%eax
8048753: 50              push   %eax
8048754: e8 17 fd ff ff   call   8048470 <fgetc@plt>
8048759: 83 c4 10         add    $0x10,%esp
804875c: 83 ec 0c         sub    $0xc,%esp
804875f: 8d 45 90         lea    -0x70(%ebp),%eax
8048762: 50              push   %eax
8048763: e8 23 ff ff ff   call   804860b <check_pass>
8048768: 83 c4 10         add    $0x10,%esp
804876b: 83 f0 01         xor     $0x1,%eax
804876e: 84 c0           test    %al,%al
8048770: 74 05           je      8048777 <main+0x72>
8048772: e8 94 fe ff ff   call   804860b <boom>
8048777: 83 ec 08         sub    $0x8,%esp
804877a: 6a 00           push   $0x0
804877c: 8d 45 88         lea    -0x78(%ebp),%eax
804877f: 50              push   %eax

0x08048745 in main ()
(gdb) nexti
0x0804874a in main ()
(gdb) nexti
0x0804874d in main ()
(gdb) nexti
0x0804874e in main ()
(gdb) nexti
0x08048750 in main ()
(gdb) nexti
0x08048753 in main ()
(gdb) nexti
0x08048754 in main ()
(gdb) nexti
Introduce la contraseña: nolase
0x08048759 in main ()
(gdb) nexti
0x0804875c in main ()
(gdb) nexti
0x0804875f in main ()
(gdb) nexti
0x08048762 in main ()
(gdb) nexti
0x08048763 in main ()
(gdb) nexti
0x08048768 in main ()
(gdb) nexti
0x0804876b in main ()
(gdb) nexti
0x0804876e in main ()
(gdb) nexti
0x08048770 in main ()
(gdb) nexti
0x08048772 in main ()
(gdb) nexti
*****
*** BOOM!!! ***
*****
[Inferior 1 (process 14237) exited with code 0377]

```

De aquí puedo deducir que la bomba se activa con una función <boom>. Dos líneas por encima podemos ver una orden test %al, %al , y en la siguiente línea se comprueba si los valores guardados en esos registros son iguales. Por lo tanto, lo único que hay que hacer es poner un breakpoint en la línea donde ocurre el test y cambiar el valor de %al para evitar ese salto:

“break *0x0804876e”

“set \$al=0”

Avanzo el programa con “cont” y veo que me pide el código, por lo tanto he conseguido evitar la contraseña.

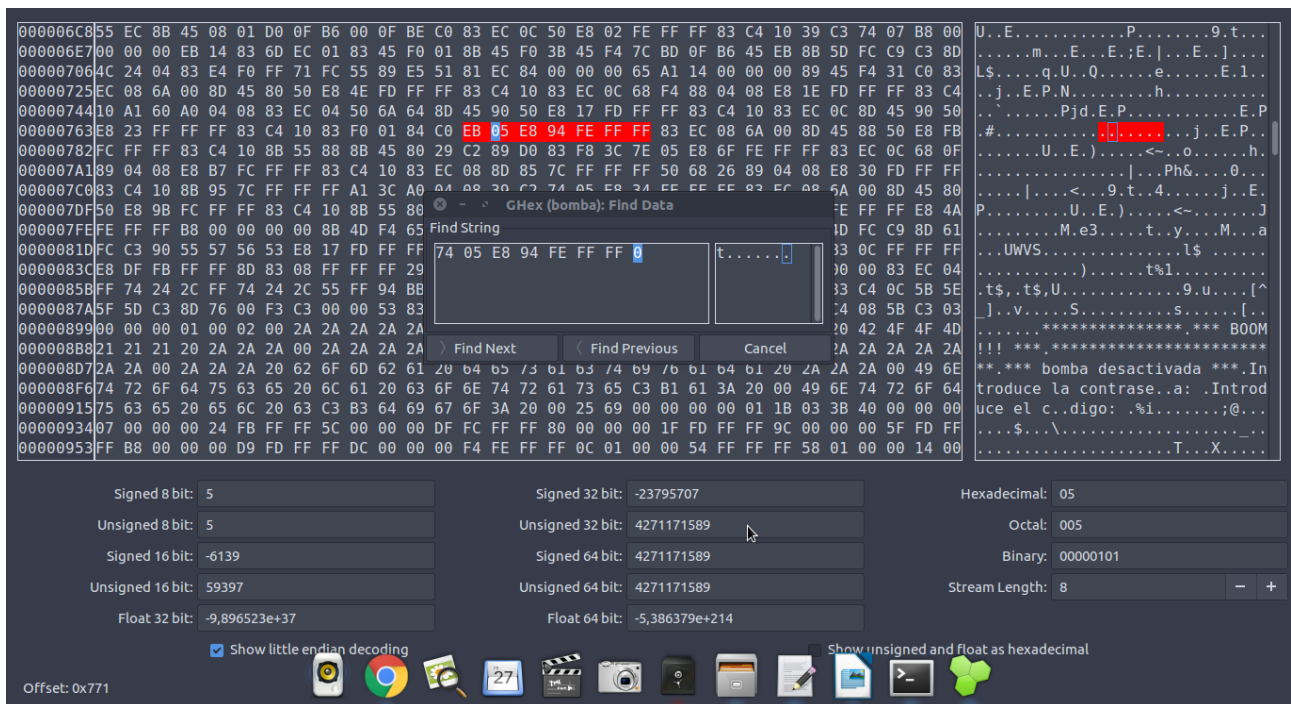
```

[Inferior 1 (process 14335) exited with code 0377]
(gdb) run
Starting program: /home/tehribbon/Escritorio/INFORMATICA/curso_recu/1ºCuatrimestre/EC/PRACTICAS/sesion4/bombas_desactivadas/Juane_garcia/bomba
Introduce la contraseña: asfgdasf

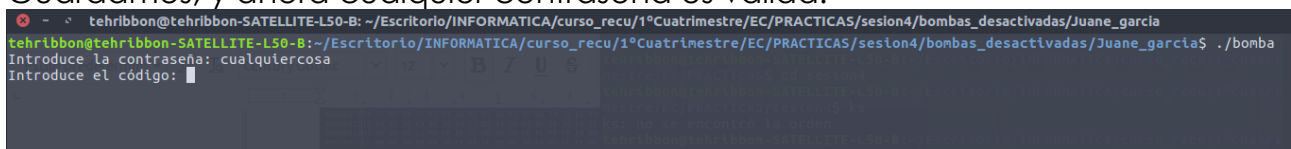
Breakpoint 2, 0x0804876e in main ()
(gdb) set $al=0
(gdb) cont
Continuando.
Introduce el código:

```

Como el salto que se produce es del tipo JE voy a cambiarlo a JMP con un editor hexadecimal, buscando la posición 74 05 e8 94 fe ff ff y cambiando 74→eb

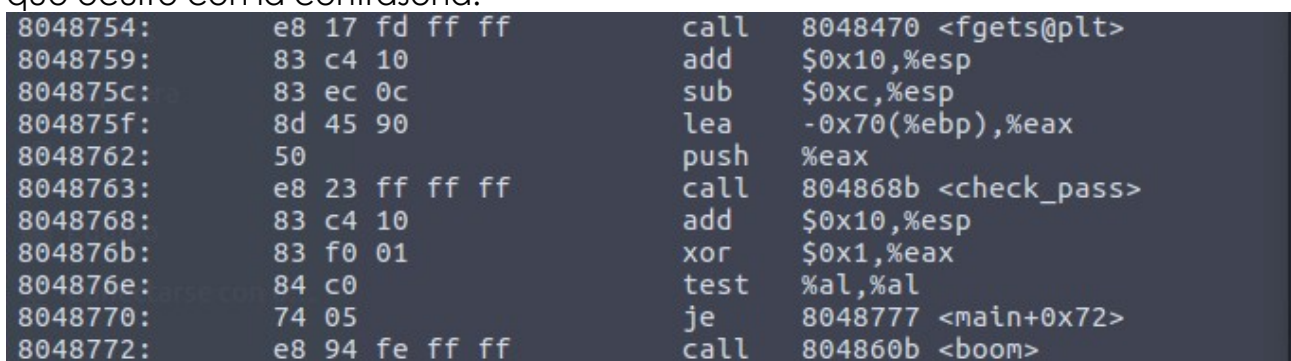


Guardamos, y ahora cualquier contraseña es válida.



Sin embargo, también puedo encontrar cual es la contraseña, procedo a explicar el proceso.

Volvemos al inicio, al punto en el que me pide la contraseña y vuelvo a observar que ocurre con la contraseña:



Veo que se llama a una función check_pass en la que probablemente se trabaje con la contraseña.

Antes de nada voy a ver las variables que hay definidas, para observar si puedo obtener algo de información:

```
0x0804a03c passcode
0x0804a040 password
```

He encontrado estas dos variables, cuyos nombres son sospechosos, (el código lo dejo para mas adelante), asique compruebo que hay en ellas:

```
(gdb) x /1sb 0x0804a040
0x0804a040 <password>: "\nLIVOM"
(gdb)
```

Dentro de la variable password hay una cadena bastante sospechosa "`\nLIVOM`". Digo sospechosa porque normalmente las cadenas o string tiene el `\n` al final y no al principio. Quizá la contraseña está invertida, es decir, se le ha dado la vuelta para corregirlo en la función `check_pass`. Si le doy la vuelta a esa cadena obtengo "`MOVIL\n`", lo cual parece una cadena válida para una contraseña. Voy a comprobar si desactiva la bomba:

```
tehrilbbon@tehrilbbon-SATELLITE-L50-B:~/Escritorio/INFORMATICA/curso_recu/1+
+Cuatrimestre/EC/PRACTICAS/sesion4/bombas_desactivadas/Juane_garcia$ ./bom
b
a
Introduce la contraseña: MOVIL
Introduce el código: 
```

Efectivamente, la contraseña es correcta. No he necesitado entrar en qué hacía la función `check_pass` porque ver el `\n` al comienzo me daba bastantes pistas, pero probablemente lo que hace esa función es darle la vuelta a la cadena.

3.CÓDIGO

```
(gdb) nexti
0x080487bb in main ()
(gdb) nexti
Introduce el código: 
```

Retomo el programa desde el último breakpoint y ejecuto "`nexti`" para ver en qué posición del se solicita la entrada del código:

```
(gdb) nexti
0x080487bb in main ()
(gdb) nexti
Introduce el código: 
```

Continúo ejecutando con "`nexti`" para ver cuando se activa la bomba:

```
0x080487bb in main ()
(gdb) nexti
Introduce el código: asdf
0x080487c0 in main ()
(gdb) nexti
0x080487c3 in main ()
(gdb) nexti
0x080487c9 in main ()
(gdb) nexti
0x080487ce in main ()
(gdb) nexti
0x080487d0 in main ()
(gdb) nexti
0x080487d2 in main ()
(gdb) nexti
*****
*** BOOM!!! ***
*****
[Inferior 1 (process 14623) exited with code 0377]
(gdb)
```

En la posición `0x080487d2` ocurre probablemente la llamada a boom, lo miro en el terminal con las instrucciones en ensamblador:


```

tehribbon@tehribbon-SATELLITE-L50-B: ~/Escritorio/INFORMATICA/curso_recu/
8048790:      89 d0      mov     %edx,%eax
8048792:      83 f8 3c   cmp     $0x3c,%eax
8048795:      7e 05      jle     804879c <main+0x97>
8048797:      e8 6f fe ff ff  call    804860b <boom>
804879c:      83 ec 0c     sub     $0xc,%esp
804879f:      68 0f 89 04 08  push    $0x804890f
80487a4:      e8 b7 fc ff ff  call    8048460 <printf@plt>
80487a9:      83 c4 10     add     $0x10,%esp
80487ac:      83 ec 08     sub     $0x8,%esp
80487af:      8d 85 7c ff ff ff  lea     -0x84(%ebp),%eax
80487b5:      50          push    %eax
80487b6:      68 26 89 04 08  push    $0x8048926
80487bb:      e8 30 fd ff ff  call    80484f0 <__isoc99_scanf@plt>
>
80487c0:      83 c4 10     add     $0x10,%esp
80487c3:      8b 95 7c ff ff ff  mov     -0x84(%ebp),%edx
80487c9:      a1 3c a0 04 08  mov     0x804a03c,%eax
80487ce:      39 c2      cmp     %eax,%edx
80487d0:      74 05      je      80487d7 <main+0xd2>
80487d2:      e8 34 fe ff ff  call    804860b <boom>
80487d7:      83 ec 08     sub     $0x8,%esp
80487da:      68 0f 89 04 08  push    $0x804890f

```

Efectivamente en esa línea se activa la bomba. Mirando las dos líneas anteriores veo que se hace un salto del tipo JE (jump equal) comparando %eax y %edx. Ahora tengo dos opciones, realizar el mismo proceso que para la contraseña (evitar ese salto), o ver que hay en esos registros para saber cual es el código correcto. Voy a optar por esta segunda opción. Pongo un punto de ruptura en la posición en la que se compara y veo que hay en esos registros:

```

(gdb) break *0x080487ce
Punto de interrupción 3 at 0x80487ce
(gdb) run
Starting program: /home/tehribbon/Escritorio/INFORMATICA/curso_recu/1ºCuatrimestre/EC/PRACTICAS/sesion4/bombas_desactivadas/Juane_garcia/bomba
Introduce la contraseña: cualquiera
Introduce el código: 978

Breakpoint 3, 0x080487ce in main ()
(gdb) info registers
eax             0x1e61      7777
ecx             0x1         1
edx             0x3d2       978
ebx             0x0         0
esp             0xffffcda0   0xffffcda0
ebp             0xffffce28   0xffffce28
esi             0xf7fac000   -134561792
edi             0xf7fac000   -134561792
eip             0x80487ce    0x80487ce <main+201>
eflags          0x28600000 [ PF SF IF ]
cs              0x23        35
ss              0x2b        43
ds              0x2b        43
es              0x2b        43
fs              0x0         0
gs              0x63        99
(gdb) x/s 0x804a03c
0x804a03c <passcode>: "a\036"
(gdb)

```

En %eax se introduce el valor del registro 0x804a03c. Con "x/s 0x804a03c" compruebo lo que hay en ese registro, que es el correspondiente a la variable "passcode". Por lo tanto, compruebo si el valor 7777 es válido para el código:

```
tehribbon@tehribbon-SATELLITE-L50-B:~/Escritorio/INFORMATICA/curso_recu/10Cuatrimestre/EC/PRACTICAS/sesion4/bombas_desactivadas/Juane_garcia$ ./bom
b
a
Introduce la contraseña: cualquiera
Introduce el código: 7777
*****
*** bomba desactivada ***
*****
tehribbon@tehribbon-SATELLITE-L50-B:~/Escritorio/INFORMATICA/curso_recu/10Cuatrimestre/EC/PRACTICAS/sesion4/bombas_desactivadas/Juane_garcia$
```

Por lo tanto, ya tenemos la bomba desactivada.