

ESTRUCTURA DE COMPUTADORES -

2º GRADO INGENIERÍA INFORMÁTICA

PRÁCTICA 4.- BOMBA DIGITAL – DESENSAMBLADORES

Mario Antonio López Ruiz - 45109755Q

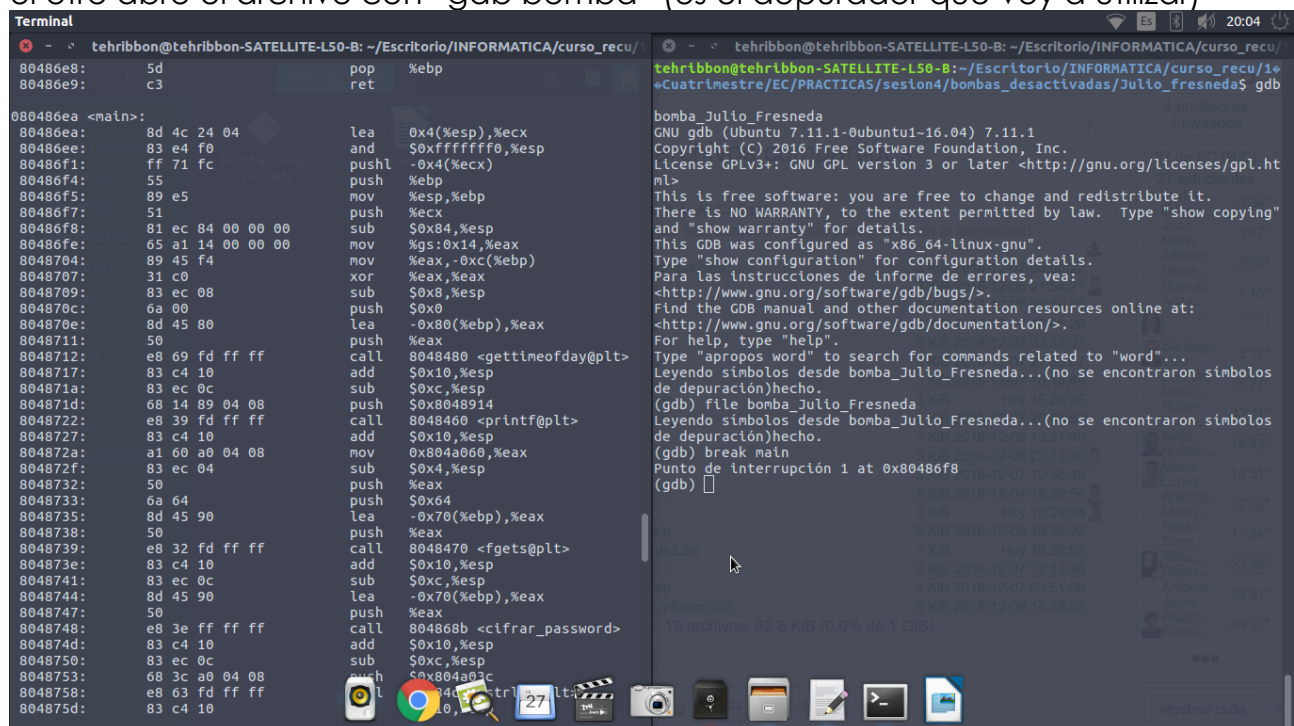
BOMBA DE JULIO ANTONIO FRESNEDA GARCÍA. COMO DESACTIVARLA

RESUMEN

En esta bomba, el autor realiza una serie de cifrados a la contraseña y al código, para intentar ocultar esa información. Sin embargo, a la hora de llamar a la función boom lo hace desde el main, con saltos del estilo JE y JLE. Lo único que hay que hacer es evitar este tipo de saltos modificándolos para que sean del tipo JMP, evitándo así las llamadas a activar la bomba. Esta modificación la realizo con un editor hexadecimal.

1.INICIO

Para comenzar abro dos terminales, en uno ejecuto "objdump -d bomba_Julio_Fresneda" 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/
80486e8: 5d                pop     %ebp
80486e9: c3                ret

080486ea <main>:
80486ea: 8d 4c 24 04       lea     0x4(%esp),%ecx
80486ee: 83 e4 f0          and     $0xfffffff0,%esp
80486f1: ff 71 fc          pushl  -0x4(%ecx)
80486f4: 55                push   %ebp
80486f5: 89 e5            mov     %esp,%ebp
80486f7: 51                push   %ecx
80486f8: 81 ec 84 00 00 00 sub     $0x84,%esp
80486fe: 65 a1 14 00 00 00 mov     %gs:0x14,%eax
8048704: 89 45 f4          mov     %eax,-0xc(%ebp)
8048707: 31 c0            xor     %eax,%eax
8048709: 83 ec 08          sub     $0x8,%esp
804870c: 6a 00            push   $0x0
804870e: 8d 45 80          lea     -0x80(%ebp),%eax
8048711: 50                push   %eax
8048712: e8 69 fd ff ff    call   8048480 <gettimeofday@plt>
8048717: 83 c4 10          add     $0x10,%esp
804871a: 83 ec 0c          sub     $0xc,%esp
804871d: 68 14 89 04 08    push   $0x8048914
8048722: e8 39 fd ff ff    call   8048460 <printf@plt>
8048727: 83 c4 10          add     $0x10,%esp
804872a: a1 60 a0 04 08    mov     0x804a060,%eax
804872f: 83 ec 04          sub     $0x4,%esp
8048732: 50                push   %eax
8048733: 6a 64            push   $0x64
8048735: 8d 45 90          lea     -0x70(%ebp),%eax
8048738: 50                push   %eax
8048739: e8 3d fd ff ff    call   8048470 <fgets@plt>
804873e: 83 c4 10          add     $0x10,%esp
8048741: 83 ec 0c          sub     $0xc,%esp
8048744: 8d 45 90          lea     -0x70(%ebp),%eax
8048747: 50                push   %eax
8048748: e8 3e fd ff ff    call   804868b <cifrar_password>
804874d: 83 c4 10          add     $0x10,%esp
8048750: 83 ec 0c          sub     $0xc,%esp
8048753: 68 3c a0 04 08    push   $0x804a03c
8048758: e8 63 fd ff ff    call   80484c0 <main+0x3c>
804875d: 83 c4 10          add     $0x10,%esp

tehrribbon@tehrribbon-SATELLITE-L50-B: ~/Escritorio/INFORMATICA/curso_recu/1*
tehrribbon@tehrribbon-SATELLITE-L50-B: ~/Escritorio/INFORMATICA/curso_recu/1*
Cuartimestre/EC/PRACTICAS/sesion4/bombas_desactivadas/Julio_fresneda$ gdb
GNU gdb (Ubuntu 7.11.1-0ubuntu1-16.04) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.ht
ml>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
Para las instrucciones de informe de errores, vea:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Leyendo símbolos desde bomba_Julio_Fresneda... (no se encontraron símbolos
de depuración)hecho.
(gdb) file bomba_Julio_Fresneda
Leyendo símbolos desde bomba_Julio_Fresneda... (no se encontraron símbolos
de depuración)hecho.
(gdb) break main
Punto de interrupción 1 at 0x80486f8
(gdb)
```

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. A partir de ahí busco en que línea se activa la bomba cuando he introducido una contraseña errónea.

```

(gdb) nexti
0x08048775 in main ()
(gdb) nexti
0x08048777 in main ()
(gdb) nexti
0x08048779 in main ()
(gdb) nexti
*****
*** BOOM!!! ***
*****
[Inferior 1 (process 15950) exited with code 0377]
(gdb) nexti
Este programa no está corriendo.
(gdb)

```

Veo que algo está ocurriendo en la posición 0x08048779 en main(), miro en el código ensamblador que está pasando:

```

8048763:      50                push    %eax
8048764:      68 3c a0 04 08      push    $0x804a03c
8048769:      8d 45 90            lea     -0x70(%ebp),%eax
804876c:      50                push    %eax
804876d:      e8 7e fd ff ff      call    80484f0 <strncmp@plt>
8048772:      83 c4 10            add     $0x10,%esp
8048775:      85 c0                test    %eax,%eax
8048777:      74 05                je      804877e <main+0x94>
8048779:      e8 8d fe ff ff      call    804860b <boom>
804877a:      83 c4 10            add     $0x10,%esp

```

Efectivamente en esa línea se está llamando a boom. Mirando dos líneas antes veo que esa llamada a activar la bomba se produce si el resultado de ejecutar test %eax, %eax cumple la condición de salto JE. Lo único que tengo que hacer ahora es modificar ese salto JE para que sea del tipo JMP. Lo realizo con un editor hexadecimal (ghex) cambiando donde aparezca 74 05 e8 8d fe ff ff → eb 05 e8 8d fe ff ff:

The screenshot shows the GHex editor with a hex dump of the assembly code. The search string 'EB 05 E8 8D FE FF 0' is entered in the Find Data dialog. The 'Find Next' button is highlighted.

Offset	Hex	ASCII
00000770	FF FF 83 C4 10 85 C0 EB 05 E8 8D FE FF	...
00000780	08 6A 00 8D 45 88 50 E8 F4 FC FF FF 83 C4 10 8B	...
00000790	55 88 8B 45 80 29 C2 89 D0 83 F8 05 7E 05 E8 68	...
000007A0	FE FF FF 83 EC 0C 68 2F 89 04 08 E8 B0 FC FF FF	...
000007B0	83 C4 10 83 EC 08 8D 85 7C FF FF FF 50 68 46 89	...
000007C0	04 08 E8 19 FD FF FF 83 C4 10 8B 85 7C FF FF FF	...
000007D0	83 EC 0C 50 E8 04 FF FF FF 83 C4 10 89 85 7C FF	...
000007E0	FF FF 8B 95 7C FF FF FF A1 48 A0 04 08 39 C2 74	...
000007F0	05 E8 15 FE FF FF 83 EC 08 6A 00 8D 45 80 50 E8	...
00000800	7C FC FF FF 83 C4 10 8B 55 80 8B 45 88 29 C2 89	...

Find String: EB 05 E8 8D FE FF 0

Find Next Find Previous Cancel

Signed 8 bit: 5 Signed 32 bit: -24254459 Hexadecimal: 05

Unsigned 8 bit: 5 Unsigned 32 bit: 4270712837 Octal: 005

Signed 16 bit: -6139 Signed 64 bit: 4270712837 Binary: 00000101

Unsigned 16 bit: 59397 Unsigned 64 bit: 4270712837 Stream Length: 8

Float 32 bit: -9,431293e+37 Float 64 bit: -5,386379e+214

Show little endian decoding Show unsigned and float as hexadecimal

Offset: 0x778

Compruebo si puedo introducir cualquier contraseña:

```
tehribbon@tehribbon-SATELLITE-L50-B:~/Escritorio/INFORMATICA/curso_recu/1Cuatrimestre/EC/PRACTICAS/sesion4/bombas_desactivadas/Julio_fresneda$ ./bomba_Julio_Fresneda
Introduce la contraseña: cualquiera
Introduce el código: 
```

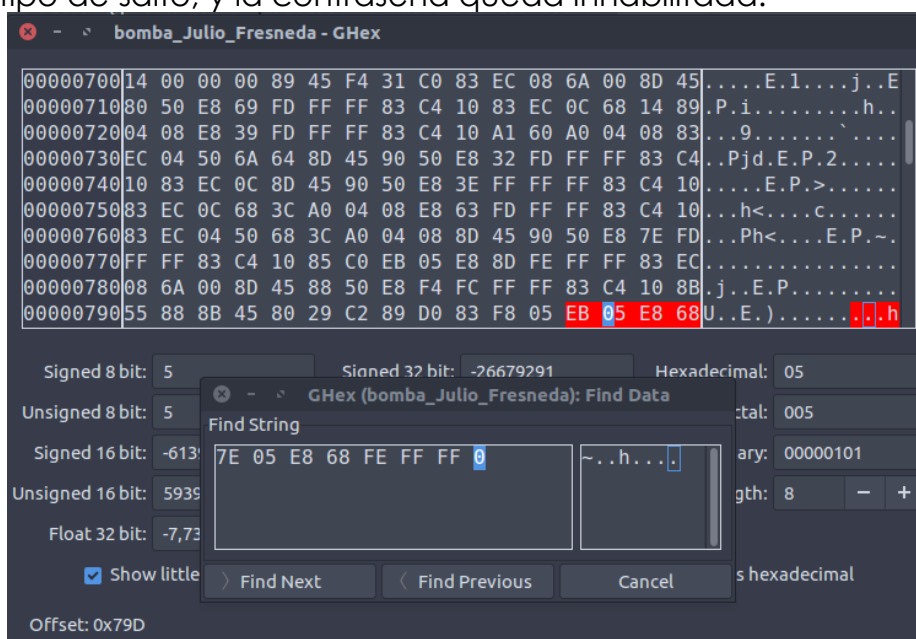
Sin embargo, si espero un tiempo e introduzco la contraseña, la bomba se activa:

```
tehribbon@tehribbon-SATELLITE-L50-B:~/Escritorio/INFORMATICA/curso_recu/1Cuatrimestre/EC/PRACTICAS/sesion4/bombas_desactivadas/Julio_fresneda$ ./bomba_Julio_Fresneda
Introduce la contraseña: s
*****
*** BOOM!!! ***
*****
tehribbon@tehribbon-SATELLITE-L50-B:~/Escritorio/INFORMATICA/curso_recu/1Cuatrimestre/EC/PRACTICAS/sesion4/bombas_desactivadas/Julio_fresneda$ 
```

Observando el código ensamblador se aprecia que se hace una comparación del tipo JLE igual que en el guión de la práctica, por lo que procedo a pasarla al tipo JMP:

804878c:	83 c4 10	add	\$0x10,%esp
804878f:	8b 55 88	mov	-0x78(%ebp),%edx
8048792:	8b 45 80	mov	-0x80(%ebp),%eax
8048795:	29 c2	sub	%eax,%edx
8048797:	89 d0	mov	%edx,%eax
8048799:	83 f8 05	cmp	\$0x5,%eax
804879c:	7e 05	jle	80487a3 <main+0xb9>
804879e:	e8 68 fe ff ff	call	804860b <boom>
80487a3:	83 ec 0c	sub	\$0xc,%esp

Cambio el tipo de salto, y la contraseña queda inhabilitada:



3.CÓDIGO

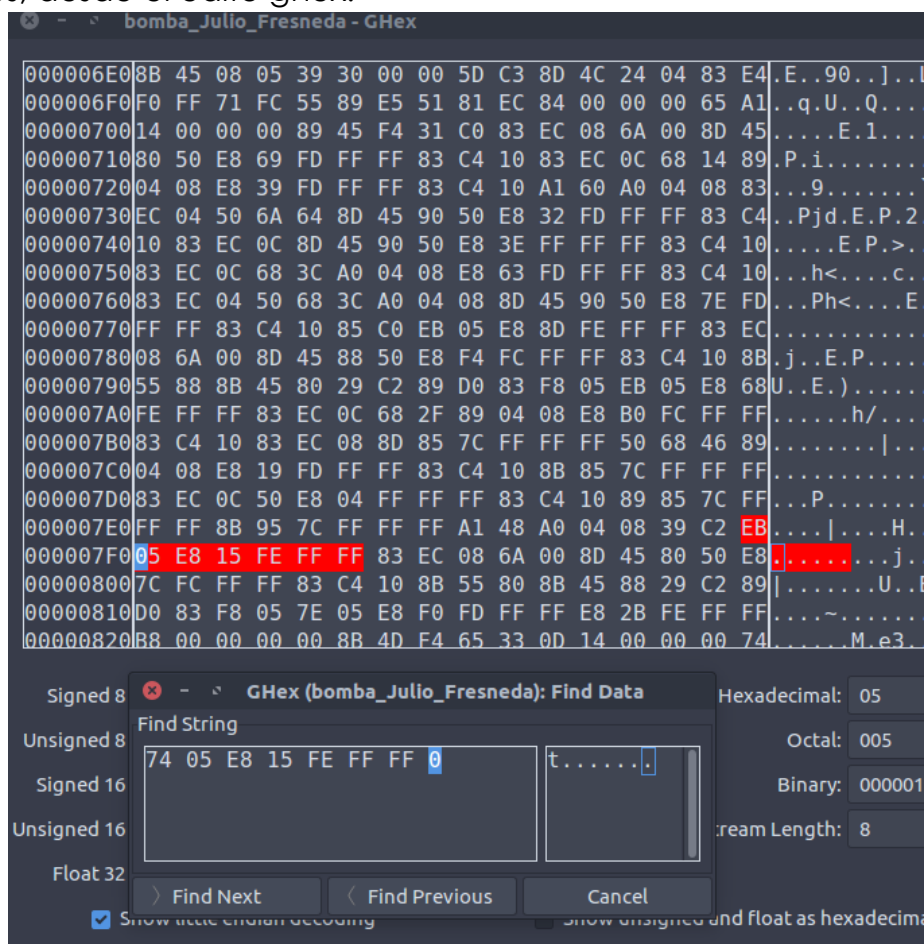
Avanzo desde el último breakpoint con "nexti" para ver donde se activa la bomba para el caso de el código, y obtengo la posición:

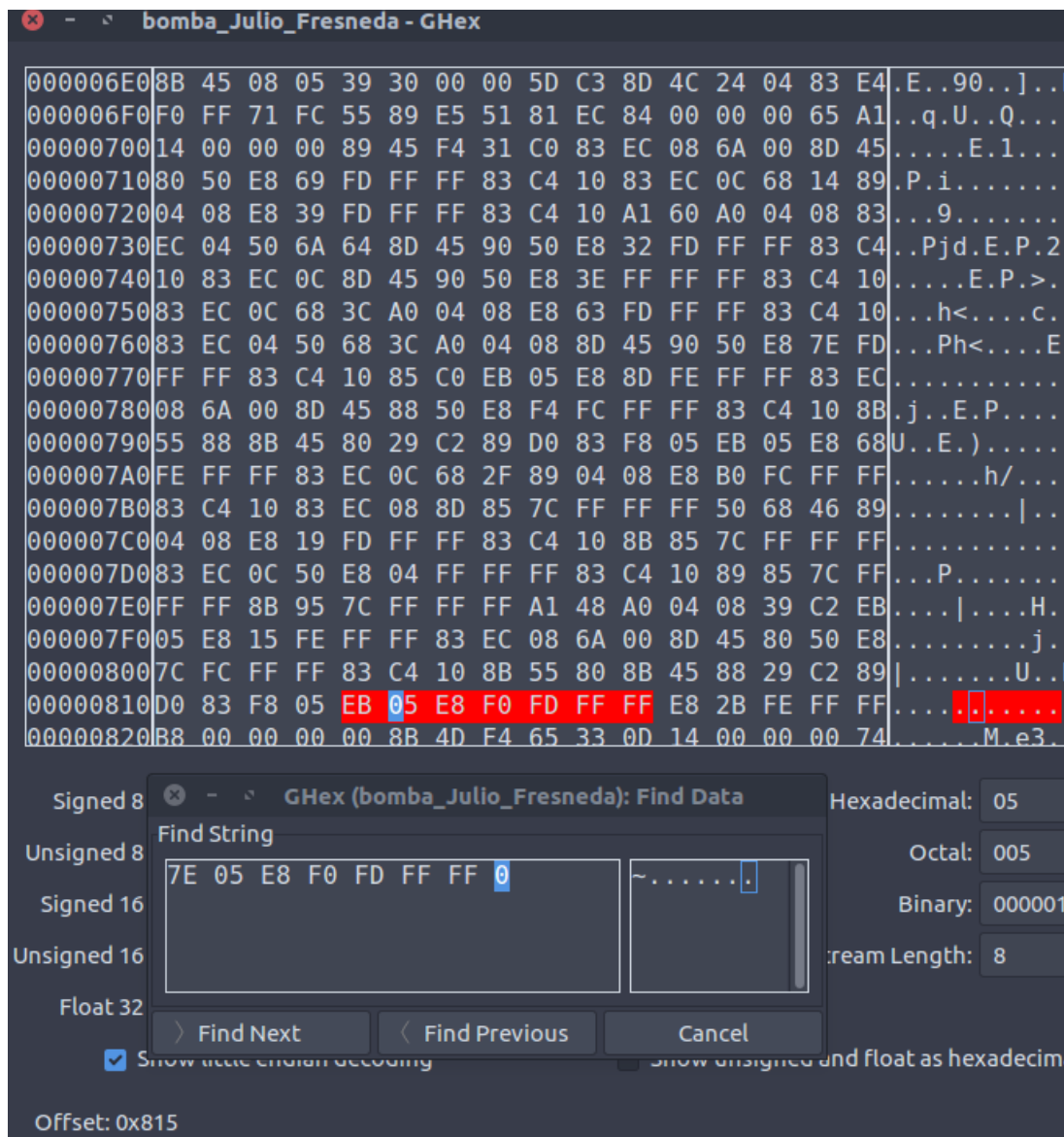
```
0x080487f1 in main ()
(gdb) nexti
*****
*** BOOM!!! ***
*****
[Inferior 1 (process 16483) exited with code 0377]
(gdb)
```

Busco en el código ensamblador:

80487ed:	39 c2	cmp	%eax,%edx
80487ef:	74 05	je	80487f6 <main+0x10c>
80487f1:	e8 15 fe ff ff	call	804860b <boom>
80487f6:	83 ec 08	sub	\$0x8,%esp
80487f9:	6a 00	push	\$0x0
80487fb:	8d 45 80	lea	-0x80(%ebp),%eax
80487fe:	50	push	%eax
80487ff:	e8 7c fc ff ff	call	8048480 <gettimeofday@plt>
8048804:	83 c4 10	add	\$0x10,%esp
8048807:	8b 55 80	mov	-0x80(%ebp),%edx
804880a:	8b 45 88	mov	-0x78(%ebp),%eax
804880d:	29 c2	sub	%eax,%edx
804880f:	89 d0	mov	%edx,%eax
8048811:	83 f8 05	cmp	\$0x5,%eax
8048814:	7e 05	jle	804881b <main+0x131>
8048816:	e8 f0 fd ff ff	call	804860b <boom>
804881b:	e8 2b fe ff ff	call	804864b <defused>

Se produce un tipo de salto JE para activar la bomba, y un poco mas abajo, en 0x08048814 se ve que se produce la misma comparación que antes del tipo JLE. Lo único que tengo que hacer es modificar esos dos saltos por JMP, al igual que hecho antes, desde el edito ghex:





Y la bomba ya estaría completamente desactivada. Comprobamos:

```

tehribbon@tehribbon-SATELLITE-L50-B: ~/Escritorio/INFORMATICA/curso_recu/
Cuatrimestre/EC/PRACTICAS/sesion4/bombas_desactivadas/Julio_fresneda$ ls
bomba_Julio_Fresneda  explicacion.odt
tehribbon@tehribbon-SATELLITE-L50-B:~/Escritorio/INFORMATICA/curso_recu/1
Cuatrimestre/EC/PRACTICAS/sesion4/bombas_desactivadas/Julio_fresneda$ ./b
o
mba_Julio_Fresneda
Introduce la contraseña: asdasdasd
Introduce el código: 1231asd
*****
*** bomba desactivada ***
*****
tehribbon@tehribbon-SATELLITE-L50-B:~/Escritorio/INFORMATICA/curso_recu/1
Cuatrimestre/EC/PRACTICAS/sesion4/bombas_desactivadas/Julio_fresneda$

```

Y efectivamente, hemos desactivado la bomba.

Ahora voy a descifrar cuales son los valores reales de la contraseña y el código.

4.DESCIFRANDO LA CONTRASEÑA

Ejecutando la orden "info variables", observo dos nombres sospechosos, "passwd" y "passwd"

```
0x0804a03c  passwd
0x0804a048  passwd
```

Voy a comprobar que albergan esas variables

```
(gdb) x/s 0x0804a03c
0x0804a03c <passwd>: "uo{eiqlf\n"
(gdb) x /1sb 0x0804a048
0x0804a048 <passwd>: "^H\001"
(gdb)
```

Como se puede observar, son dos cadenas que no parecen tener mucho sentido. La primera acaba en \n, que es el carácter de terminación de una cadena. Si pruebo esta cadena en el programa, la bomba se activa.

```
tehribbon@tehribbon-SATELLITE-L50-B:~/Escritorio/INFORMATICA/curso_recu/1ºCuatri
mestre/EC/PRACTICAS/sesion4/bombas_desactivadas/Julio_fresneda$ ./bomba_Julio_Fr
esneda
Introduce la contraseña: uo{eiild
*****
*** BOOM!!! ***
*****
tehribbon@tehribbon-SATELLITE-L50-B:~/Escritorio/INFORMATICA/curso_recu/1ºCuatri
mestre/EC/PRACTICAS/sesion4/bombas_desactivadas/Julio_fresneda$
```

Por lo tanto, es probable que esas cadenas hayan sufrido algún tipo de transformación para ocultar su verdadero valor.

Busco en la funcion main en que punto se pide la contraseña (el mismo proceso que en el punto 1) y veo que ocurre después:

```
(gdb) nextt
0x08048779 in main ()
(gdb) nexti
*****
*** BOOM!!! ***
*****
```

Busco el punto 0x08048779 en ensamblador:

8048739:	e8 32 fd ff ff	call	8048470 <fgets@plt>
804873e:	83 c4 10	add	\$0x10,%esp
8048741:	83 ec 0c	sub	\$0xc,%esp
8048744:	8d 45 90	lea	-0x70(%ebp),%eax
8048747:	50	push	%eax
8048748:	e8 3e ff ff ff	call	804868b <cifrar_password>
804874d:	83 c4 10	add	\$0x10,%esp
8048750:	83 ec 0c	sub	\$0xc,%esp
8048753:	68 3c a0 04 08	push	\$0x804a03c
8048758:	e8 63 fd ff ff	call	80484c0 <strlen@plt>
804875d:	83 c4 10	add	\$0x10,%esp
8048760:	83 ec 04	sub	\$0x4,%esp
8048763:	50	push	%eax
8048764:	68 3c a0 04 08	push	\$0x804a03c
8048769:	8d 45 90	lea	-0x70(%ebp),%eax
804876c:	50	push	%eax
804876d:	e8 7e fd ff ff	call	80484f0 <strncmp@plt>
8048772:	83 c4 10	add	\$0x10,%esp
8048775:	85 c0	test	%eax,%eax
8048777:	74 05	je	804877e <main+0x94>
8048779:	e8 8d fe ff ff	call	804860b <boom>

Entre que se pide la contraseña (en 0x08048739) y explota la bomba (0x08048779) ocurren 3 cosas destacables, se llama a tres funciones, "cifrar_password", "[strlen@plt](#)" y "[strncmp@plt](#)". Teniendo en cuenta la cadena que obtuvimos antes, "uo{eiql d\n", probablemente lo que está ocurriendo es que esta cadena se está transformando en "cifrar_password" y después con "[strncmp@plt](#)" se compara esta cadena, que será la contraseña original, con la que nosotros hemos introducido.

Para comprobar esta hipótesis, voy a estudiar paso a paso que ocurre en "cifrar_password". Coloco un punto de ruptura en esa función con "break cifrar_password"

0804868b <cifrar_password>:			
804868b:	55	push	%ebp
804868c:	89 e5	mov	%esp,%ebp
804868e:	83 ec 18	sub	\$0x18,%esp
8048691:	c7 45 f4 00 00 00 00	movl	\$0x0,-0xc(%ebp)
8048698:	eb 26	jmp	80486c0 <cifrar_password+0x

Vemos que comienza el programa y da un salto a 80486c0:

80486c0:	83 ec 0c	sub	\$0xc,%esp
80486c3:	68 3c a0 04 08	push	\$0x804a03c
80486c8:	e8 f3 fd ff ff	call	80484c0 <strlen@plt>
80486cd:	83 c4 10	add	\$0x10,%esp
80486d0:	8d 50 ff	lea	-0x1(%eax),%edx
80486d3:	8b 45 f4	mov	-0xc(%ebp),%eax
80486d6:	39 c2	cmp	%eax,%edx
80486d8:	77 c0	ja	804869a <cifrar_password+0x
80486da:	90	nop	
80486db:	c9	leave	
80486dc:	c3	ret	

Aquí vemos una posible estructura de bucle, si se da el caso de que %eax y %edx cumplen la condición de JA (jump above), es decir, cuando eax es mayor que edx.

Por ello vamos a comprobar que valores se están guardando en esos registros.

Se está guardando el valor de 0x804a03c para mandarlo como argumento a [strlen@plt](#), por lo que voy a ver que se guarda en esa posición:

```
0x080486c8 in cifrar_password ()
(gdb) x /1sb 0x804a03c
0x804a03c <password>: "uo{eiqlf\n"
(gdb)
```

Vemos que coincide con la cadena hallada anteriormente, por lo que vamos por buen camino.

En el valor de edx se está guardando el tamaño de la cadena sospechosa, que es 9, y en eax se está guardando el valor 0:

```
eax          0x0          0
ecx          0xf7fb600c    -134520820
edx          0x9          9
```

Por lo tanto, como se está produciendo la condición de salto, nos encontramos en un bucle, que embarca:

```
804869a: 8b 45 f4      mov     -0xc(%ebp),%eax
804869d: 83 e0 01      and     $0x1,%eax
80486a0: 85 c0         test    %eax,%eax
80486a2: 75 18         jne     80486bc <cifrar_password+0x1>
80486a4: 8b 55 f4      mov     -0xc(%ebp),%edx
80486a7: 8b 45 08      mov     0x8(%ebp),%eax
80486aa: 01 d0         add     %edx,%eax
80486ac: 8b 4d f4      mov     -0xc(%ebp),%ecx
80486af: 8b 55 08      mov     0x8(%ebp),%edx
80486b2: 01 ca         add     %ecx,%edx
80486b4: 0f b6 12      movzbl  (%edx),%edx
80486b7: 83 c2 05      add     $0x5,%edx
80486ba: 88 10         mov     %dl,(%eax)
80486bc: 83 45 f4 01   addl    $0x1,-0xc(%ebp)
80486c0: 83 ec 0c      sub     $0xc,%esp
80486c3: 68 3c a0 04 08 push   $0x804a03c
80486c8: e8 f3 fd ff ff call    80484c0 <strlen@plt>
80486cd: 83 c4 10      add     $0x10,%esp
80486d0: 8d 50 ff      lea     -0x1(%eax),%edx
80486d3: 8b 45 f4      mov     -0xc(%ebp),%eax
80486d6: 39 c2         cmp     %eax,%edx
80486d8: 77 c0         ja      804869a <cifrar_password+0x1>
```

Produzco otra vez la iteración completa y observo que edx se mantiene constante, y que eax aumenta en 1:

```
(gdb) info registers
eax          175      0x1          1
ecx          176      0xf7fb600c    -134520820
edx          177      0x9          9
ebx          178      0x0          0
```

Cuando llega a la iteración numero 9 se sale del bucle y ejecuta "nop" y vuelve al main:

```
eax          175      0x9          9
ecx          176      0xf7fb600c    -134520820
edx          177      0x9          9
```

```
(gdb) nexti
0x080486dc in cifrar_password ()
(gdb) nexti
0x0804874d in main ()
```


Compruebo cual es la cadena que se compara antes de llamar a strcmp:

```
804876c: 50          push    %eax
804876d: e8 7e fd ff ff  call    80484f0 <strncmp@plt>
```

Y al comprobar que hay en eax:

```
(gdb) x/s 0xffffcd98
0xffffcd98: "soqoxe\017"
(gdb)
```

Obtengo "soqoxe\017", una cadena que parece albergar algún sentido, pero con variaciones.

Tras probar con distintas contraseñas (con "hola" obtengo "moqa\017", con "holahola" obtengo "moqamoqa\017"), deduzco que se está aplicando alguna operación a la cadena introducida mediante ascii, al carácter de posiciones pares se les suma 5 en su valor ascii y a los impares se les deja igual:

n → 110 + 5 → 115 → s
o → → → → → → → o
l → 108 + 5 → 113 → q
o → → → → → → → o
s → 115 + 5 → 120 → x
e → → → → → → → e

Ahora, teniendo en cuenta que obtuve una cadena que no tenía sentido "uo{eiiqlf", y que a la cadena que yo introduzco se le suma 5 a los pares y los impares se deja igual, hago el proceso inverso:

u → 117 - 5 → 112 → p
o → → → → → → → o
{ → 123 - 5 → 118 → v
e → → → → → → → e
i → 105 - 5 → 100 → d
i → → → → → → → i
q → 113 - 5 → 108 → l
l → → → → → → → l
f → 102 - 5 → 97 → a

He obtenido "povedilla" como solución, compruebo si es válida:

```
tehribbon@tehribbon-SATELLITE-L50-B: ~/Escritorio/INFORMATICA/curso_recu/1ºCuatri  
mestre/EC/PRACTICAS/sesion4/bombas_desactivadas/Julio_fresneda$ ./bomba_Julio_Fr  
esneda  
Introduce la contraseña: povedilla  
Introduce el código: 
```

Y efectivamente, hemos hallado la contraseña.

5.DESCIFRANDO EL CÓDIGO

Realizo lo mismo que con la contraseña, veo que ocurre desde que se pide el código hasta que se activa la bomba:

```
80487c2: e8 19 fd ff ff      call 80484e0 <__isoc99_scanf@plt>
80487c7: 83 c4 10            add    $0x10,%esp
80487ca: 8b 85 7c ff ff ff   mov    -0x84(%ebp),%eax
80487d0: 83 ec 0c            sub    $0xc,%esp
80487d3: 50                  push   %eax
80487d4: e8 04 ff ff ff     call 80486dd <cifrar_passcode>
80487d9: 83 c4 10            add    $0x10,%esp
80487dc: 89 85 7c ff ff ff   mov    %eax,-0x84(%ebp)
80487e2: 8b 95 7c ff ff ff   mov    -0x84(%ebp),%edx
80487e8: a1 48 a0 04 08      mov    0x804a048,%eax
80487ed: 39 c2               cmp    %eax,%edx
80487ef: 74 05              je     80487f6 <main+0x10c>
80487f1: e8 15 fe ff ff     call 804860b <boom>
```

Veó que se ejecuta una función "cifrar_passcode":

```
080486dd <cifrar_passcode>:
80486dd: 55                  push   %ebp
80486de: 89 e5              mov    %esp,%ebp
80486e0: 8b 45 08           mov    0x8(%ebp),%eax
80486e3: 05 39 30 00 00     add    $0x3039,%eax
80486e8: 5d                 pop    %ebp
80486e9: c3                 ret
```

Esta función le suma el valor de \$0x3039 a eax, que es el valor 12345. Por lo tanto se está sumando el valor introducido por pantalla + 12345.

Sigo en el main y veo una comparación, edx con eax, voy a comprobar que valores contienen ambos:

```
(gdb) nexti
0x080487ed in main ()
(gdb) info registers
eax             0x1485e    84062
ecx             0x1        1
edx             0x303a     12346
```

Habiendo introducido como código "1", el programa le ha sumado "12345" → "12346". Lo está comparando con el valor 84062, el cual, probablemente sea el valor a conseguir.

Por lo tanto realizo $84062 - 12345 = 71717$

Compruebo si este código funciona:

```
tehribbon@tehribbon-SATELLITE-L50-B:~/Escritorio/INFORMATICA/curso_recu/1ºCuatri
mestre/EC/PRACTICAS/sesion4/bombas_desactivadas/Julio_fresneda$ ./bomba_Julio_Fr
esneda
Introduce la contraseña: povedilla
Introduce el código: 71717
*****
*** bomba desactivada ***
*****
```

Ya hemos desactivado la bomba por completo.