

BOMBA – JULIO FRESNEDA

49215154F – juliofresnedag@correo.ugr.es

Cómo averiguar la contraseña y entender el algoritmo de encriptación.

→ Abrimos el ejecutable desde la terminal con objdump -d. Vemos que hay un método que encripta la contraseña. Lo intentamos entender:

```
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+0x35>
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+0x31>
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+0xf>
80486da: 90                nop
80486db: c9                leave
80486dc: c3                ret
```

Si se sigue paso a paso, se entiende que es un bucle for, desde 0 hasta la longitud de la contraseña introducida -1. Ésto se ve en las órdenes:

```
movl $0x0, -0xc(%ebp)    // Carga el i del for, y lo iguala a 0
jmp 80486c0 <cifrar_password+0x35>    // Salta a esa dirección
```

Desde 80486c0 hasta `cmp %eax, %edx` completa el funcionamiento del for

Dentro del for:

```
mov -0xc(%ebp), %eax; and $0x1,%eax; test %eax, %eax    // If(i %2 == 0)
jne 80486bc <cifrar_password+0x31>    // Si i%2 != 0 salta a esa dirección
En caso contrario, hace pass[i] = pass[i] + 5 en    add $0x5,%edx
```

En resumen, se puede deducir que hace `pass[i] = pass[i] + 5` en las posiciones pares de la contraseña. Ya sabemos el algoritmo, ahora hay que saber la contraseña encriptada para invertir su encriptación. Sabemos que `strlen` usa la contraseña encriptada para comparar el tamaño. Vamos a buscarla desde la terminal, con `ddd`.

The screenshot shows a debugger window with a menu bar (File, Edit, View, Program, Commands, Status, Source, Data) and a toolbar with icons for Lookup, Find, Break, Watch, and Print. The main window displays assembly code for a function, with addresses ranging from 0x0804873e to 0x08048797. The code includes instructions like `add $0x10,%esp`, `sub $0xc,%esp`, `lea -0x70(%ebp),%eax`, `push %eax`, `call 0x804868b <cifrar_password>`, `add $0x10,%esp`, `sub $0xc,%esp`, `push $0x804a03c`, `call 0x80484c0 <strlen@plt>`, `add $0x10,%esp`, `sub $0x4,%esp`, `push %eax`, `push $0x804a03c`, `lea -0x70(%ebp),%eax`, `push %eax`, `call 0x80484f0 <strncmp@plt>`, `add $0x10,%esp`, `test %eax,%eax`, `je 0x804877e <main+148>`, `call 0x804860b <boom>`, `sub $0x8,%esp`, `push $0x0`, `lea -0x78(%ebp),%eax`, `push %eax`, `call 0x8048480 <gettimeofday@plt>`, `add $0x10,%esp`, `mov -0x78(%ebp),%edx`, `mov -0x80(%ebp),%eax`, `sub %eax,%edx`, and `mov %edx,%eax`.

A dialog box titled "DDD: Examine Memory" is open, showing "Examine 10" in a text field, "char" in a dropdown menu, "bytes" in a dropdown menu, and "from 0x804a03c" in a text field. The dialog has buttons for "Print", "Display", "Close", and "Help".

At the bottom of the debugger window, the following memory dump is visible:

```
0x804a044 <password+8>: 102 'f' 10 '\n'
(gdb) x /10cb 0x804a03c
0x804a03c <password>: 117 'u' 111 'o' 123 '{' 101 'e' 105 'i' 105 'i' 113 'q' 108 'l'
0x804a044 <password+8>: 102 'f' 10 '\n'
(gdb) |
```

The status bar at the bottom left shows "A03c".

Vemos que justo antes de llamar a strlen, se hace push de una dirección. Vemos que contiene con Data → Memory. Abajo se muestra la etiqueta <password>. Hemos acertado, ahí está la contraseña. Cogemos los caracteres que salen hasta \n, y nos sale la contraseña encriptada “uo{eiiqlf”. Conociendo el algoritmo, vamos a descriptarla.

u - 5 = p
{ - 5 = o
i - 5 = d
q - 5 = l
f - 5 = a

El resultado es “povedilla”. Ésa es la contraseña.

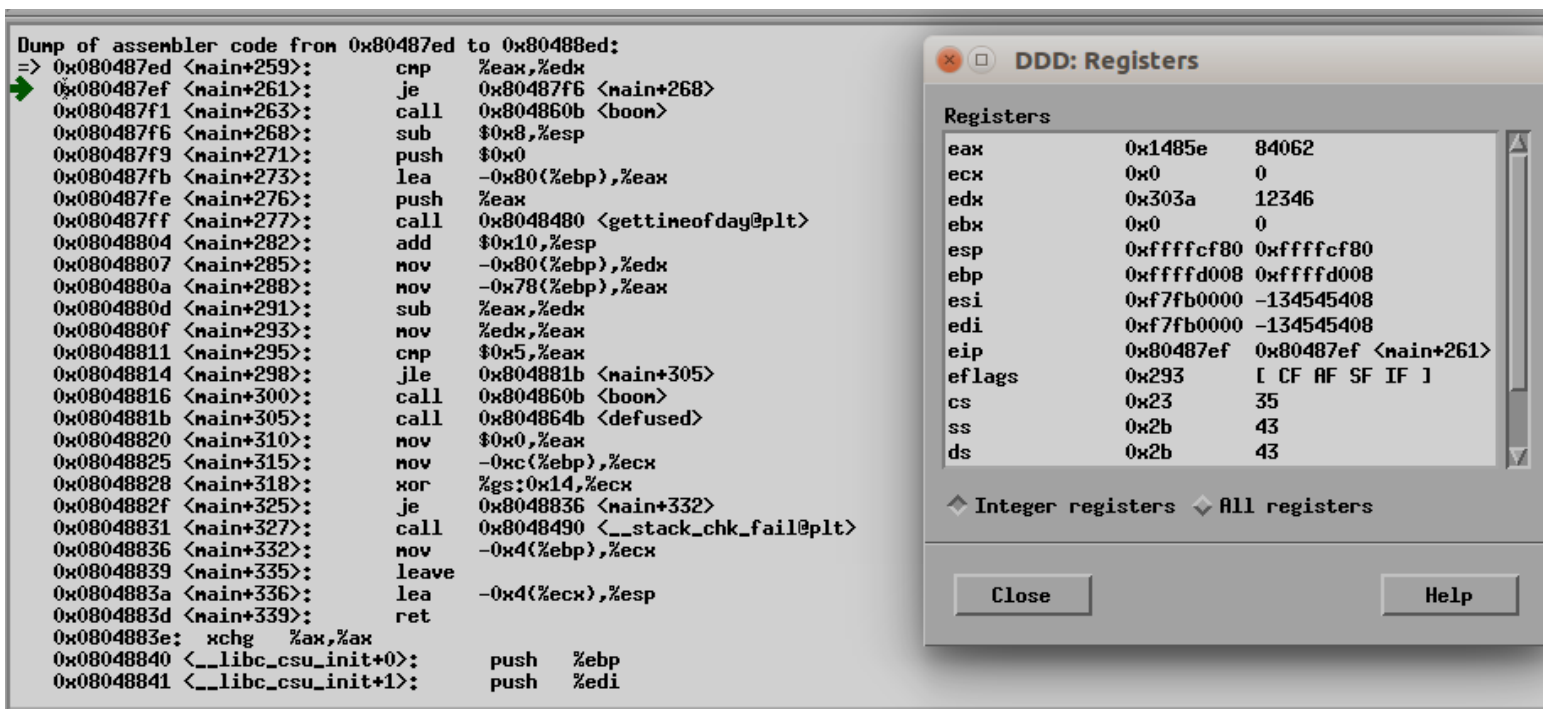
Cómo averiguar el passcode y entender el algoritmo

Igual que antes, abrimos la bomba con objdump -d y observamos el método de encriptación del 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
```

La encriptación es bastante sencilla, simplemente se le suma al passcode el número hexadecimal 3039, 12345 en decimal.

Vamos a buscar el passcode encriptado con ddd.



The screenshot shows a debugger window with two panes. The left pane displays assembly code from address 0x080487ed to 0x080488ed. The right pane shows the state of the CPU registers.

Assembly Code:

```
Dump of assembler code from 0x080487ed to 0x080488ed:
=> 0x080487ed <main+259>:    cmp     %eax,%edx
0x080487ef <main+261>:    je      0x080487f6 <main+268>
0x080487f1 <main+263>:    call   0x0804860b <boom>
0x080487f6 <main+268>:    sub     $0x8,%esp
0x080487f9 <main+271>:    push    $0x0
0x080487fb <main+273>:    lea     -0x80(%ebp),%eax
0x080487fe <main+276>:    push    %eax
0x080487ff <main+277>:    call   0x08048480 <gettimeofday@plt>
0x08048804 <main+282>:    add     $0x10,%esp
0x08048807 <main+285>:    mov     -0x80(%ebp),%edx
0x0804880a <main+288>:    mov     -0x78(%ebp),%eax
0x0804880d <main+291>:    sub     %eax,%edx
0x0804880f <main+293>:    mov     %edx,%eax
0x08048811 <main+295>:    cmp     $0x5,%eax
0x08048814 <main+298>:    jle     0x0804881b <main+305>
0x08048816 <main+300>:    call   0x0804860b <boom>
0x0804881b <main+305>:    call   0x0804864b <defused>
0x08048820 <main+310>:    mov     $0x0,%eax
0x08048825 <main+315>:    mov     -0xc(%ebp),%ecx
0x08048828 <main+318>:    xor     %gs:0x14,%ecx
0x0804882f <main+325>:    je      0x08048836 <main+332>
0x08048831 <main+327>:    call   0x08048490 <__stack_chk_fail@plt>
0x08048836 <main+332>:    mov     -0x4(%ebp),%ecx
0x08048839 <main+335>:    leave
0x0804883a <main+336>:    lea     -0x4(%ecx),%esp
0x0804883d <main+339>:    ret
0x0804883e:    xchg    %ax,%ax
0x08048840 <__libc_csu_init+0>:    push    %ebp
0x08048841 <__libc_csu_init+1>:    push    %edi
```

Registers:

Register	Value (Hex)	Value (Dec)
eax	0x1485e	84062
ecx	0x0	0
edx	0x303a	12346
ebx	0x0	0
esp	0xffffcf80	0xffffcf80
ebp	0xffffd008	0xffffd008
esi	0xf7fb0000	-134545408
edi	0xf7fb0000	-134545408
eip	0x080487ef	0x080487ef <main+261>
eflags	0x293	[CF AF SF IF]
cs	0x23	35
ss	0x2b	43
ds	0x2b	43

Justo después de la llamada a scanf y la llamada al método de encriptar, la bomba compara el passcode introducido ya encriptado con el passcode encriptado correcto. Para eso usa `cmp %eax,%edx`. Si vemos el registro en éste paso, vemos que `%eax` tiene el valor `0x1485e`, 84062 en decimal.

Ahora vamos a restarle 12345 a ese número, dando como resultado 71717, el passcode correcto.

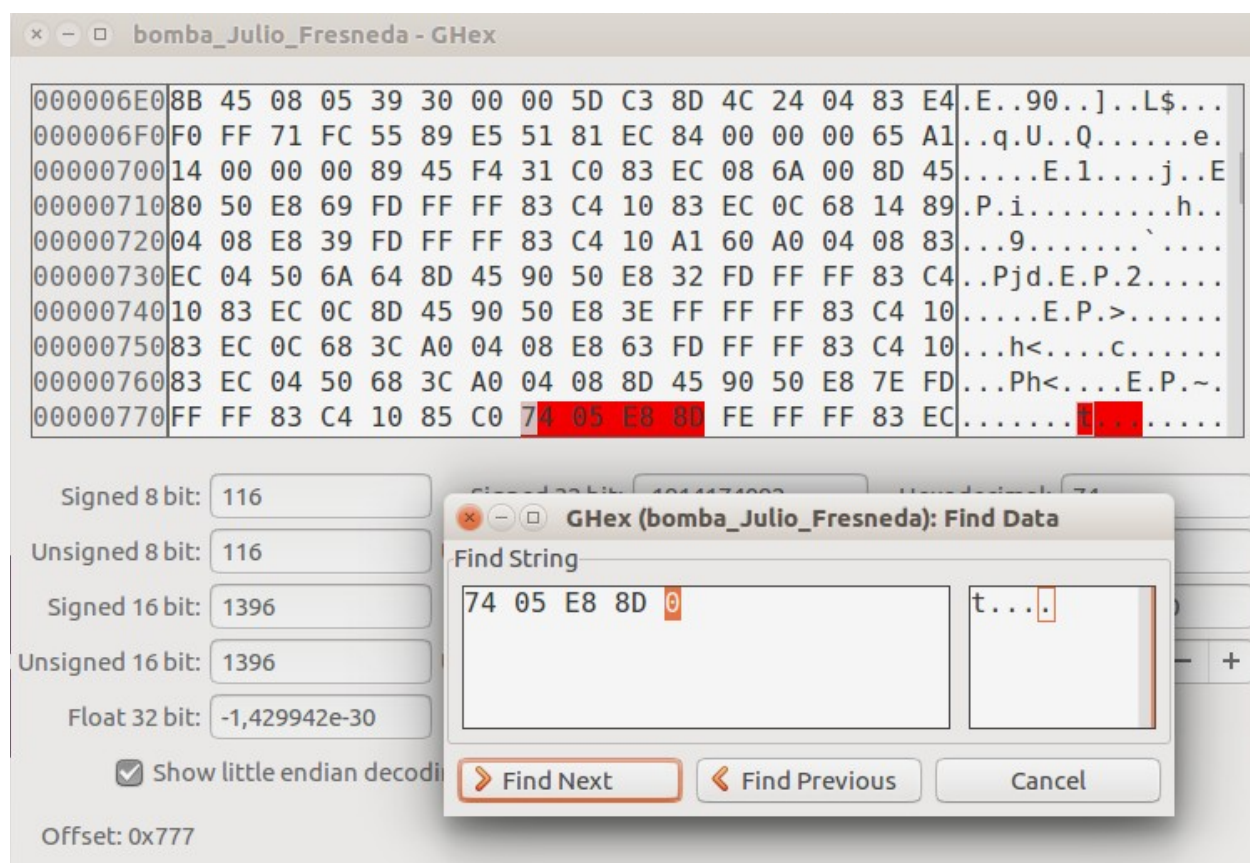
Ya tenemos la contraseña y el passcode.

Cómo desactivar la bomba.

Abriendo la bomba con obdump -d , se puede ver que antes de cada llamada a <boom> se hace una comprobación, donde se usa je. Desactivarla es tan fácil como transformar ese je en un jmp.

```
8048777: 74 05 je 804877e <main+0x94>
8048779: e8 8d fe ff ff call 804860b <boom>
80487ef: 74 05 je 80487f6 <main+0x10c>
80487f1: e8 15 fe ff ff call 804860b <boom>
```

Para ésto se va a modificar la bomba con la herramienta de edición hexadecimal ghex. Je equivale a 74 en hexadecimal, y jmp equivale a eb. Vamos a buscar esas dos series de números hexadecimales en ghex, y editarlos.



Una vez cambiado el 74 por EB, se puede ejecutar la bomba y no explotará, pongamos la contraseña y código que queramos.

```
julioxxx@julio-pc:~/Escritorio$ ./bomba_Julio_Fresneda
Introduce la contraseña: eee
Introduce el código: 111
*****
*** bomba desactivada ***
*****
```