

Desactivación Bomba de María Camarero

Estructura de Computadores - Grupo C3

Mario Rodríguez Ruiz

1. Contraseña

Para averiguar la contraseña se ha utilizado el depurador **DDD** realizando los siguientes pasos:

En primer lugar se ha puesto un punto de ruptura en la llamada a **fgets**, que es cuando se le pide al usuario por pantalla que ingrese una contraseña.

Una vez funcionando el programa y detenido ahí se ha metido una contraseña al azar para avanzar.

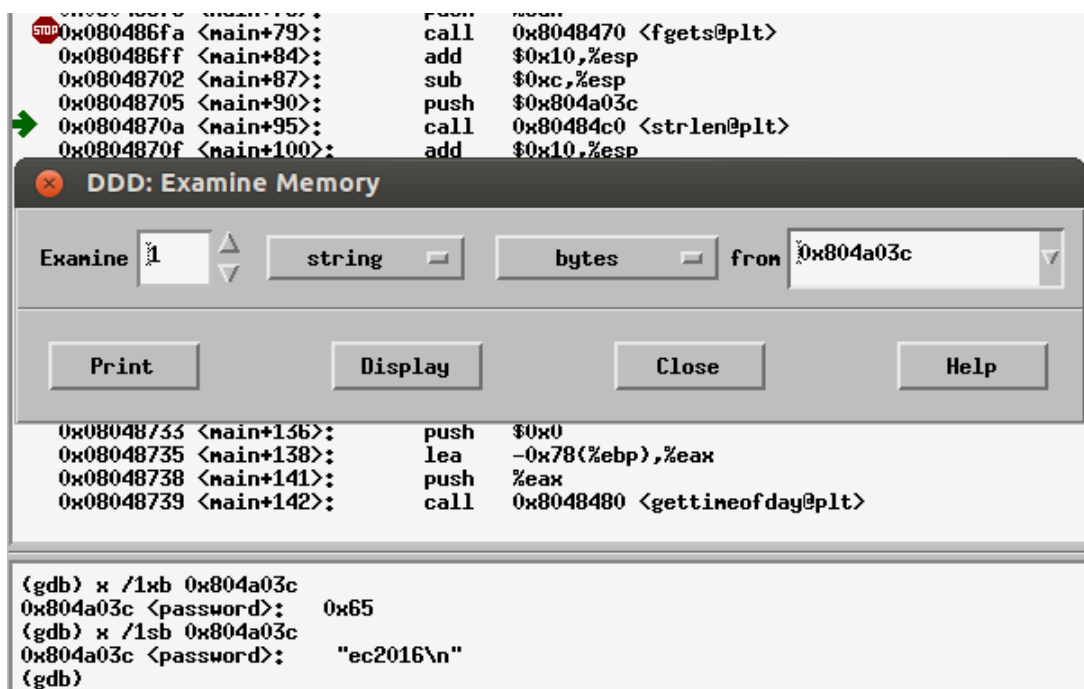


Figura 1.1: Examine Memory desde DDD

Justo después del guardado de la contraseña introducida, se hace un **push** "sospechoso". Ha sido el momento de examinar los datos en memoria mediante la opción **Data** → **Memory**: Poniendo Examine a 1, tipo **string** en **bytes** desde la dirección **0x804a03c** y haciéndole un **print** ha mostrado la contraseña tal y como se ve en la Figura 1.1.

Contraseña: **ec2016**

2. Código

Para averiguar la contraseña se ha utilizado el depurador **DDD** realizando los siguientes pasos:

Llegando a la sección del programa que se encarga de procesar y validar el código, se ha puesto un punto de ruptura en esta parte para futuros intentos.

Como se ve en la Figura 2.1 el programa hace una llamada a una función con nombre **desactivar**, por lo que mediante **stepi** la ejecución seguirá su curso adentrándose en ésta.

```
0x08048795 <nain+234>:    sub    %eax,%eax
0x08048797 <nain+236>:    nov    %edx,%eax
0x0804879a <nain+239>:    cmp    $0x3c,%eax
0x0804879c <nain+241>:    jle    0x80487a1 <nain+246>
0x0804879c <nain+241>:    call   0x804860b <boom>
0x080487a1 <nain+246>:    nov    -0x84(%ebp),%eax
0x080487a7 <nain+252>:    sub    $0xc,%esp
0x080487aa <nain+255>:    push   %eax
0x080487ab <nain+256>:    call   0x804868b <desactivar>
0x080487b0 <nain+261>:    add    $0x10,%esp
0x080487b3 <nain+264>:    call   0x804864b <defused>
0x080487b8 <nain+269>:    nov    $0x0,%eax
End of assembler dump.

Introduce la contraseña: ec2016
Introduce el código: 1111

Breakpoint 2, 0x08048797 in main ()
(gdb)
```

Figura 2.1: Examine Memory desde DDD para el código

Una vez dentro de dicha función, tal y como se muestra en la Figura 2.2, se pone un punto de ruptura para futuros intentos y a continuación se examina.

```
Dump of assembler code for function desactivar:
=> 0x0804868b <+0>:    push   %ebp
0x0804868c <+1>:    nov    %esp,%ebp
0x0804868e <+3>:    sub    $0x8,%esp
0x08048691 <+6>:    nov    0x8(%ebp),%eax
0x08048694 <+9>:    lea    -0x7e0(%eax),%edx
0x0804869a <+15>:    nov    0x804a068,%eax
0x0804869f <+20>:    cmp    %eax,%edx
0x080486a1 <+22>:    je     0x80486a8 <desactivar+29>
0x080486a3 <+24>:    call   0x804860b <boom>
0x080486a8 <+29>:    nop
0x080486a9 <+30>:    leave
0x080486aa <+31>:    ret
End of assembler dump.
```

Figura 2.2: Examine Memory desde DDD para el código

Mediante la ayuda del estado de los registros (**Status**→**Registers**) y avanzando una a una cada instrucción se puede ver cómo van modificando sus valores.

El código de prueba introducido ha sido **1111**, sin embargo "extrañamente" se ha visto cómo se ha modificado su valor convirtiéndose ahora en **-905**, como puede verse en la Figura 2.3. Es ahora cuando se hace una comparación de este nuevo valor con el valor que contiene **%eax** que es cero.

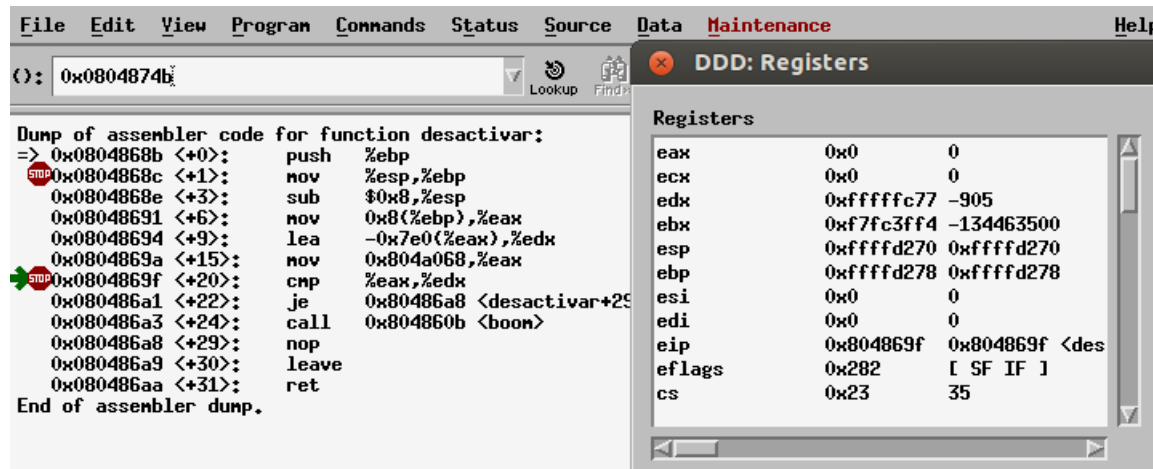


Figura 2.3: Examine Memory desde DDD para el código

Aquí es cuando se aprecia que se ha realizado una resta sobre el valor introducido y que su resultado tiene que ser cero. Por lo que sumando el valor introducido como prueba (1111) y el resultado después de la modificación (905) aparece la solución al problema: $1111+905=2016$

Por tanto, el código es **2016**.