`ED` > **Prácticas** > **Práctica 2**

3️⃣ Desensamblado

> Alumno: Javier Gómez López <javi5454@correo.ugr>

> Fecha: 28 de noviembre, 2021.

> Curso: 2º Doble Grado en Ingeniería Informática y Matemáticas, 2021/2022.

Bomba personal

Mi bomba pide que introduzcas dos cosas:

- Una **clave** (númerica), generada **aleatoriamente** cada vez que se ejecuta la bomba.
- Una contraseña (alfanumérica), que tiene como valor `"estoestrivial\n"`.

Además, se han introducido en el código algunas funciones para *despistar*.

- `distract()` que devuelve exactamente el mismo número que se le ha pasado como argumento. Dentro además usamos un `while` que hace de contador, pero cuya utilidad es nula. Por otro lado, a la hora de tratar de desensamblar el binario, añade confusión adicional que se pase a esta función una variable declarada pero que no se inicializa (su valor es aleatorio y se encuentra en memoria).

```
int distraction;

...

if(distract(distraction) == distraction){
    ...
}
```

este `if` siempre se cumplirá y ejecutará su código.

- `credits()` imprime por pantalla información sobre el creador de la bomba.

Por otro lado, en el `main()` se genera el código aleatorio. El código de mi bomba es el siguiente.

💻 Código de `JGL_Bomba_2021.c`

```
// gcc -Og JGL_Bomba_2021.c -o bombajaviergomez -no-pie -fno-guess-branch-
probability

#include <stdio.h> // printf(),fgets(), scanf()
#include <stdlib.h> // exit()
#include <string.h> //strncmp()
#include <sys/time.h> // gettimeofday(), struct timeval

#define SIZE 100
#define TLIM 10
#define MAXSIZE 100

char password[]="mambaout\n"; //contrasenia
```

```c
void credits(){ //Muestra el autor de la bomba
    printf("\n Bomba por Javier Gómez López\n");
    printf("    @javi5454 - github.com/Javi5454\n");
    printf("    2º DGIIM 2021/2022, UGR\n");
}

void boom(void){ // Imprime BOOM! si no se adivina la contraseña o se acaba el
tiempo
    printf("\n"
           "***************\n"
           "*** BOOM!!! ***\n"
           "***************\n"
           "\n");

           credits();

    exit(-1);
}

void defused(void){ //Imprime bomba desactivada si se desactiva la bomba
    printf("\n"
           ".........................\n"
           "··· bomba desactivada ···\n"
           ".........................\n"
           "\n");

           credits();

    exit(0);
}

int distract(int n){ //Función de distracion, no tiene utilidad real

    int seguir = 0;

    int i = 0;

    while(i < 10 && seguir == 0){
        i++;
    }

    return n;
}

int main(){
    char password[]="estoestrivial\n";
    int code;
    struct timeval tv1, tv2;

    int enter_code;
    char enter_password[MAXSIZE];

    int distraction;

    srand(time(NULL));
    code = rand() % 100;

    gettimeofday(&tv1, NULL);
```

```
    printf("Rápido, tienes 10 segundos, inserta la [%i] ", code);

    if(distract(distraction) == distraction){
        printf("clave: ");
    }

    scanf("%i", &enter_code);

    gettimeofday(&tv2, NULL);

    if(tv2.tv_sec - tv1.tv_sec > TLIM){
        boom();
    }

    if( enter_code != code ){
        boom();
    }

    printf("\nAhora introduce la contraseña\n");
    printf("¿Quieres una pista? (Y/N) ");
    fgets(enter_password, MAXSIZE, stdin);
    fgets(enter_password, MAXSIZE, stdin);

    if(strcmp(enter_password, "Y") == 0 || strcmp(enter_password, "Y\n") == 0){
        printf("De acuerdo: Frase más repetida en matemáticas\n");
    }
    else{
        printf("Okey, vas fuerte eh\n");
    }

    gettimeofday(&tv1, NULL);

    printf("Introduce la contraseña (Tienes 10 segundos): ");
    fgets(enter_password, MAXSIZE, stdin);

    gettimeofday(&tv2, NULL);

    if(strcmp(enter_password, password) != 0 || tv2.tv_sec - tv1.tv_sec > TLIM){
        boom();
    }
    else{
        defused();
    }

}
```

Desactivando mi propia bomba

1. Compilación del programa

   Para compilar el programa simplemente hay que copiar el comando del principio del código fuente `JGL_Bomba_2021.c`. El comando es el siguiente:

   ```
   gcc -Og JGL_Bomba_2021.c -o bombajaviergomez -no-pie -fno-guess-branch-probability
   ```

2. Evitar las comprobaciones

   En caso de fallo gráfico de gdb, pulsar ***Control+L***

   Primero compilamos el programa con `gcc`. Una bez compilado, lo ejecutamos paso a paso con `gdb` para evitar las comprobaciones. El comando es el siguiente:

   ```
   gdb -tui bombajaviergomez
   ```

   Se nos abrirá `gdb` y mostramos el código ensamblador y los registros de memoria.

   ```
   layout asm
   layout regs
   ```

   Establecemos ahora un *breakpoint* en la función `main` para comenzar la depuración.

   ```
   br main
   run
   ```

   Ahora deseamos avanzar en la ejecución del código. Para ello, ejecutamos la orden

   ```
   si
   ```

   Podemos presionar *ENTER* tras un comando para repetir el mismo. Si observamos el código ensamblador, vemos que hay llamadas a funciones como `time` o `srand` o `rand`. Por ello establecemos otro *breakpoint* más adelante y continuamos con la ejecución.

   Los primeros dos *callq* nos muestran lo siguiente

   ```
   <gettimeofday@plt>
   <__printf_chk@plt>
   ```

   Sin embargo, el siguiente es más interesante pues muestra

   ```
   <distract>
   ```

   lo que nos puede dar una pista de que ese código es inútil y está para despistar.

   Si continuamos con nuestro código, observamos un *cpm* tras un segundo `gettimeofday()`, por lo cual esta es la primera comprobación temporal y debemos establecer ahí un *breakpoint*

   ```
   br *main+188
   ```

   LLegados a este punto, o introducimos un código cualquiera más rápido de la cuenta, o establecemos *eax* a 0 para así cumplir el tiempo siempre.

   ```
   set $eax=0
   ```

   Si seguimos avanzando con si, vemos que llegamos a otro *cpm*, esta vez tenemos

   ```
   cmp     %eax, 0xc(%rsp)
   ```

En *eax* tenemos el valor de nuestro código en ese momento. Ahora tenemos dos opciones: o bien volvemos a ejecutar el código sabiendo donde podemos comprobar el código, o establecemos *eax* a el valor del código que hayamos introducido nosotros. En este caso, el código es `2804`.

```
set $eax=<valor introducido>

ó

p (int)$eax
```

Si continuamos leyendo el código ensamblador, tenemos varias funciones `fgets()`, pero antes de otra llamada a `getttimeofday()`, lo cual nos da indicios de que son de distracción.

Más adelante encontramos un `strcmp()`, y tras el, un `test()`. Establecemos un *breakpoint* en el test.

```
br *main+480
```

En este punto, podemos ver los argumentos que se le han pasado a `strcmp()`

```
x/s $rdi
x/s $rsi
```

De esta manera, veremos nuestra contraseña introducida y la contraseña que es, este caso, `"estoestrivial\n"`. Para saltar esta comprobacion tenemos

```
set $eax=0
```

haciendo que el test se salte la bomba.