Práctica 4. Estructura de Computadores Noelia Escalera Mejías. Grupo A3

Apartado 1: Cómo se ha codificado la bomba y código fuente

La contraseña de mi bomba es spmg, no obstante ésta está codificada en ASCII (11511210910310 si incluimos el \n). El pin es 2007, pero este no está codificado. He aquí el código fuente de mi programa:

```
// gcc -Og bomba_NoeliaEscaleraMejias.c -o bomba_NoeliaEscaleraMejias -no-pie
#include <stdio.h>
                     // para printf(), fgets(), scanf()
#include <stdlib.h> // para exit()
#include <string.h> // para strncmp()
#include <sys/time.h> // para gettimeofday(), struct timeval
#define SIZE 100
#define TLIM 5
char password[]="11511210910310"; // contraseña codificada
int passcode = 2007;
                                            // pin
void boom(void){
       printf("\n"
"********
       "*** BOOM!!! ***\n"
       "*****\n"
       "\n");
exit(-1);
void defused(void){
       printf("\n"
        "·····\n"
       "·····\n"
       "\n");
       exit(0);
char * codifica (const char * c){
       char * a_devolver = malloc(sizeof(char) * (strlen(c) * 4 +1));
       char * nueva_c = a_devolver;
       *nueva_c = \sqrt{0};
       for (;*c; ++c){
       nueva_c += sprintf(nueva_c, "%u ", *c);
       return a_devolver;
```

```
int main(){
       char pw[SIZE];
       int pc, n;
       struct timeval tv1,tv2;
                                      // gettimeofday() secs-usecs
       gettimeofday(&tv1,NULL);
               printf("\nIntroduce la contraseña: ");
       while (fgets(pw, SIZE, stdin) == NULL );
       char * cad = codifica (pw);
       if
              (strncmp(cad,password,sizeof(password)) )
           boom();
       gettimeofday(&tv2,NULL);
             ( tv2.tv_sec - tv1.tv_sec > TLIM )
           boom();
       do { printf("\nIntroduce el pin: ");
  if ((n=scanf("%i",&pc))==0)
       scanf("%*s")
                        ==1;
       while (n!=1);
             (pc != passcode )
           boom();
       gettimeofday(&tv1,NULL);
             ( tv1.tv_sec - tv2.tv_sec > TLIM )
           boom();
       defused();
```

Apartado 2: Cómo descodificar la bomba

Nos metemos en gdb con el ejecutable bomba_NoeliaEscaleraMejias y avanzamos con ni hasta que nos pida la constraseña. Introducimos una contraseña aleatoria, hemos introducido "hola".

```
0x7fffff7dcfa00
0x7fffffffdb10
rsi
                  0x64
                             100
                                                                                                            0x0 0
0x246 582
0x7fffffffdc60
                  0x0 0
0x602010 6299664
                                                                                                                                140737488346208
                  0x4006e0 4196064
                                                                                                                       0
[ PF IF ]
                  0x4008bd 0x4008bd <main+81>
                                                                                                            0x206
                                         0x24f(%rip),%rsi # 0)
$0x1,%edi
$0x0,%eax
0x4006a0 <_printf_chk@plt>
0x30(%rsp),%rdi
0x2007e8(%rip),%rdx
                                                                        # 0x400aec
      <400896 <main+42>
      40089d <main+49>
      <4008a2 <main+54>
                                 callq
lea
mov
      4008ac <main+64>
                                                                            # 0x6010a0 <stdin@@GLIBC_2.2.5>
                                 PC: 0x4008bd
          00004008a7 in main ()
 (000000000004008ac in main ()
gdb) ni
x000000000004008b1 in main ()
 (000000000004008b8 in main ()
 (000000000004008bd in main ()
  roduce la contraseña: hola
```

Si seguimos avanzando, vemos que se va a llamar a una función codificar, vamos a comprobar que se le pasa a la función. Vamos a ver qué se le pasa como argumento:

```
0x7fffffffdb10
                                                   140737488345872

        0x0
        0

        0x7ffff7dd18d0
        140737351850192

        0x7fffffffdb10
        140737488345872

                                                 174156911
140737488345872
                      0xa616c6f
0x7fffffffdb10
                                                                                                                     rdi
                      0x4009a0 0x4009a0 <__libc_csu_init>
0x602675 6301301
                                                                                                                                               0x7ffffffffdae0
0x7fffff7fda500
                                                                                                                                                                           0x7ffffffffdae0
140737353983232
                      0x602010 6299664
0x4006e0 4196064
                                                                                                                                               0x246 582
0x7fffffffdc60
                                                                                                                                                                           140737488346208
                                                                                                                                                             0
[ IF ]
                      0x4008cc 0x4008cc <main+96>
                                                     0x30(%rsp),%rdi
0x2007e8(%rip),%rdx
$0x64,%esi
0x400680 <fgets@plt>
   0x4008b1 <main+69>
0x4008b8 <main+76>
0x4008bd <main+81>
                                          mov
mov
callq
                                                                                                   # 0x6010a0 <stdin@GLIBC 2.2.5>
                                                     %rax,%rax
0x400896 <main+42>
                                          lea 0x30(%rsp),%rdi
callq 0x4007fb <codifica>
   0x4008cc <main+96
                                                                                                    # 0x601080 <password
                                                                                                                                                                                                                           PC: 0x4008cc
         .
000004008c2 in main ()
     00000004008c5 in main ()
000000000004008c7 in main ()
000000000004008cc in main ()
db) x/1sb $rdi
7fffffffdb10: "hola\n"
```

Efectivamente, se le ha pasado la contraseña que hemos introducido. Posteriormente se llama a una función strncmp para comparar dos cadenas, analizamos los argumentos:

```
0x602a80 6302336
                    0x0 0
0x601080 6295680
                                                                                                                                   20
                                                                                                  rdi
                                                                                                                       0x602a80 6302336
                    0x4009a0 0x4009a0 < libc csu init>
                                                                                                                       0x7fffffffdae0
                                                                                                                       0x400aea 4197098
0x7fffffffdc60
                                                                                                  r11
r13
                    0x0
                    0x4006e0 4196064
                                                                                                                                              140737488346208
                    0 \times 0
                    0x4008e0 0x4008e0 <main+116
                                                                                                  eflags
    0x4008c2 <main+86>
                                             0x400896 <main+42>
0x30(%rsp),%rdi
0x4007fb <codifica>
$0x14,%edx
   0x4008c5 <main+89>
0x4008c7 <main+91>
                                    je
lea
   0x4008cc <main+96>
0x4008d1 <main+101>
   0x4008d6 <main+106>
0x4008dd <main+113>
                                              0x2007a3(%rip),%rsi
%rax,%rdi
                                                                                   # 0x601080 <password>
    0x4008e0 <main+116
                                    callq 0x400650 <strncmp@plt>
    0x4008e7 <main+123>
                                              0x4008ee <main+130>
itive process 13840 In: main
                                                                                                                                                                               L?? PC: 0x4008e0
    .
000000004008dd in main ()
 000000000004008e0 in main ()
    ) x/1sb rsi
mbolo ∎rsi∎ no est∎ en el contexto actual.
gdb) x/1sb %rsi
gdb) x/1sb %rsi
x601080 <password>: "115 112 109
gdb) x/1sb %rdi
"104 111 108 97 10 "
                             "115 112 109 103 10 "
```

El argumento password (se supone que la contraseña real de la bomba) es "115 112 109 103 10". Se le compara con "104 111 108 97 10", sin embargo, esta no es la contraseña que nosotros hemos introducido, es

la contraseña codificada. Vamos a reanudar la depuración, pero esta vez solo introduciremos un carácter (la a) como contraseña para ver qué le hace. Volvemos a chequear los argumentos de srtncmp:

```
ox602a80 6302336
                 0x0 0
0x601080 6295680
                                                                                                    0x602a80 6302336
                                                                                  rdi
                 0x4009a0 0x4009a0 <__libc_csu_init>
                 0 \times 0
                                                                                                    0x0 0
0x400aea 4197098
0x7fffffffdc60
                 0x4006e0 4196064
                                                                                                                       140737488346208
                 0x4008e0 0x4008e0 <main+116>
                                      0x400896 <main+42>
0x30(%rsp),%rdi
0x4007fb <codifica>
                              je
lea
callq
   0x4008c5 <main+89>
   0x4008cc <main+96>
   0x4008d1 <main+101>
0x4008d6 <main+106>
                              mov
lea
                                       $0x14,%edx
0x2007a3(%rip),%rsi
                                                                      # 0x601080 <password>
                                       %rax,%rdi
                                      0x400650 <strncmp@plt>
                               callq
               nain+116
                                                                                                                                                  L?? PC: 0x4008e0
   .
0000000004008dd in main ()
gdb) ni
      0000004008e0 in main ()
```

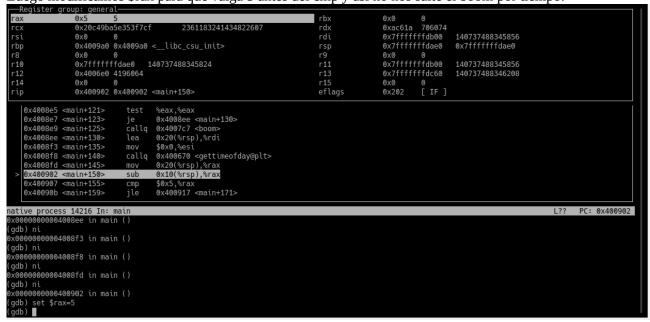
Obviamente la contraseña original del programa es la misma, pero ahora nuestra contraseña se ha traducido como "97 10". 97 es el equivalente ASCII de 'a' y 10 el de '\n'. Luego la codificación consiste en hacer el ascii de cada carácter de la contraseña y separarlo por un espacio, luego la contraseña del programa es "spmg\n".

Seguimos avanzando en nuestro programa (modificando los registros convenientes para que no nos salte el boom): primero modificamos %rsi para que valga lo mismo que %rdi y %rax antes del test, así hacemos que %eax valga 0 y es como si hubiéramos introducido bien la contraseña.

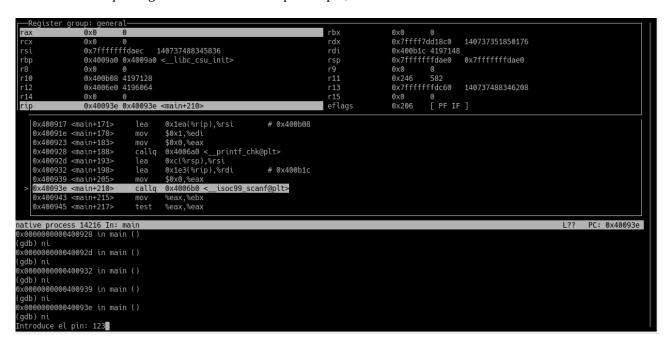
```
rdx
                                                                                                                 0x602a80 6302336
0x7ffffffffdae0
                                                                                                                                      0x7fffffffdae0
                    0x4009a0 0x4009a0 < libc csu init:
                                                                                              rsp
                                                                                                                  0x0 0
0x400aea 4197098
                    0x4006e0 4196064
                                                                                                                                       140737488346208
                                                                                                                             0
[ PF ZF IF ]
                    0x4008e0 0x4008e0 <main+116>
                                                                                                                  0x246
    0x4008c2 <main+86>
0x4008c5 <main+89>
                                           %rax,%rax
0x400896 <main+42>
                                   je
lea
callq
                                           0x30(%rsp),%rdi
0x4007fb <codifica>
    0x4008c7 <main+91>
    0x4008d1 <main+101>
0x4008d6 <main+106>
                                            $0x14,%edx
0x2007a3(%rip),%rsi
                                   mov
lea
                                                                                # 0x601080 <password>
                                             %rax,%rdi
                                                   0650 <strncmp@plt>
                                   callq 0x40
                  main+116>
                                            %eax,%eax
0x4008ee <main+130>
native process 14216 In: main
                                                                                                                                                                      L?? PC: 0x4008e0
   smbolo rsi no est en el contexto actual.
b) x/1sb $rsi
 (601080 <password>: '
gdb) x/1sb $rdi
(602a80: "97 10 "
                             "115 112 109 103 10 "
   b) set $rsi=0x60302336
            $rsi=60302336
```

Nota: Los dos primeros set son accidentales

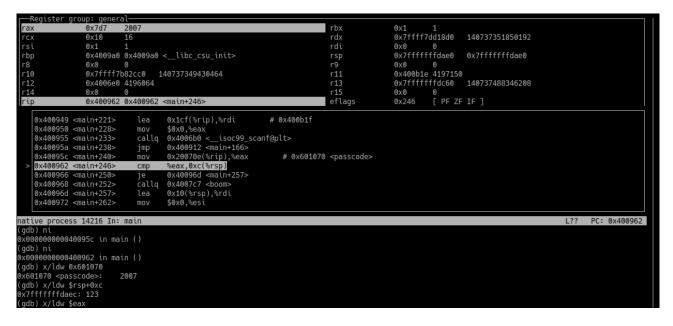
Luego modificamos \$rax para que valga 5 antes del cmp y así no nos salte el boom por tiempo:



Ahora tenemos que llegar hasta donde se nos pide el pin, nosotros introduciremos 123.



Avanzamos hasta que llegamos a una instrucción cmp a la que se le pasa como argumento un passcode. Comprobamos los argumentos de la operación:



Efectivamente, uno de los argumentos es nuestro pin, el otro argumento es 2007, que suponemos que es el pin de la bomba. El pin no está codificado.

Ya podemos salir del depurador, vamos a ejecutar el programa para comprobar que hemos hallado los datos correctos:

```
noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/EC/Prácticas/Práctica_4$ ./bomba_NoeliaEscaleraMejias
Introduce la contraseña: spmg
Introduce el pin: 2007
.....bomba desactivada ...
```

Como podemos observar, los datos eran correctos y hemos desactivado la bomba.