ESTRUCTURA DE COMPUTADORES
2ºGRADO INGENIERÍA INFORMÁTICA
PRÁCTICA 4.- BOMBA DIGITAL - DESENSAMBLADORES

HALLAR CONTRASEÑA Y CÓDIGO

1.CONTRASEÑA

Una vez desactivada la bomba, voy a explicar como encontrar cual era la contraseña y el código para la misma.

Al desactivar, la bomba, obtuvimos que en el punto de la función CmpContrs donde se activa la misma:

```
80487d8:       83 ec 0c             sub    $0xc,%esp
80487db:       68 3c a0 04 08       push   $0x804a03c
80487e0:       e8 fb fc ff ff       call   80484e0 <strlen@plt>
80487e5:       83 c4 10             add    $0x10,%esp
80487e8:       89 c2                mov    %eax,%edx
80487ea:       8b 45 ec             mov    -0x14(%ebp),%eax
80487ed:       83 ec 04             sub    $0x4,%esp
80487f0:       52                   push   %edx
80487f1:       50                   push   %eax
80487f2:       ff 75 d4             pushl  -0x2c(%ebp)
80487f5:       e8 16 fd ff ff       call   8048510 <strncmp@plt>
80487fa:       83 c4 10             add    $0x10,%esp
80487fd:       85 c0                test   %eax,%eax
80487ff:       74 05                je     8048806 <CmpContrs+0x15b>
8048801:       e8 25 fe ff ff       call   804862b <boom>
```

Observando un poco el código antes de activar la bomba, se observa que se producen llamadas a strlen@plt y a strncmp@plt, la primera función calcula la longuitud de un string y la segunda compara dos strings.
Llegados a este punto voy a echar un vistazo a las variables que hay en el programa a ver si puedo deducir algo.
Pongo un breakpoint en 80487db y comienzo a investigar desde ahí:
Al ejecutar "info variables" obtengo dos variables sospechosas con las etiquetas:

```
0x0804a03c  password
0x0804a048  passcode
```

Voy a acceder a esas posiciones de memoria para ver si obtengo algo:

```
(gdb) x /1sb 0x0804a03c
0x804a03c <password>:   "adavitcased"
(gdb)
```

Obtengo una cadena bastante sospechosa "adavitcased", ejecuto el programa para introducir esta contraseña y ver si es correcta, pero la bomba se activa.
Continuando con el código en ensamblador de antes vemos que justo antes de de ejecutar strlen se hace un "push $0x0804a03c", posicion de memoria que coincide con nuestra cadena sospechosa.
Seguimos leyendo en ensamblador y vemos que se ejecuta strcmp@plt en un punto, función que compara dos cadenas, probablemente comparando la cadena introducida por pantalla y la real para activar la bomba o no.
Llego hasta ese punto y compruebo los valores que se le han pasado como argumento a esa función, obtengo:

```
0x080487fd in CmpContrs ()
(gdb) info registers
eax            0x1      1
ecx            0x64     100
edx            0xffffcdc8       -12856
ebx            0xffffcd60       -12960
esp            0xffffcd50       0xffffcd50
ebp            0xffffcd98       0xffffcd98
esi            0xf7fac000       -134561792
edi            0xf7fac000       -134561792
eip            0x80487fd        0x80487fd <CmpContrs+338>
eflags         0x286    [ PF SF IF ]
cs             0x23     35
ss             0x2b     43
ds             0x2b     43
es             0x2b     43
fs             0x0      0
gs             0x63     99
(gdb)
```

Después de llamar a la función, parece ser que se guarda un valor en esp. En esp tenemos la posicion de memoria 0xffffcd50, por lo tanto voy a mirar que valor guarda:

```
(gdb) x /1sb 0xffffcd50
0xffffcd50:     "desactivada"
(gdb)
```

Esta cadena tiene un poco mas de sentido ya como contraseña, ejecuto de nuevo el programa para comprobar si es correcta y:

```
tehribbon@tehribbon-SATELLITE-L50-B:~/Escritorio/INFORMATICA/curso_recu/1ºCuatri
mestre/EC/PRACTICAS/sesion4/Ficheros_fuente$ ./bomba_Mario_Lopez_Ruiz
Introduce la contraseña: desactivada
Introduce el código:
```

Y efectivamente, ya hemos encontrado el valor de la contraseña.

2.CÓDIGO

Como ya tenemos el valor de la contraseña, vamos a recorrer el main hasta que nos pidan el código, y vemos que ocurre paso a paso:

```
0x0804891e in main ()
(gdb) nexti
0x08048923 in main ()
(gdb) nexti
Introduce el código:
```

En 0x0804891e se está produciendo la llamada para pedir el código por pantalla, vamos al código en ensamblador y vemos que ocurre:

```
804891e:    e8 dd fb ff ff      call   8048500 <__isoc99_scanf@plt
>
8048923:    83 c4 10            add    $0x10,%esp
8048926:    8b 85 7c ff ff ff   mov    -0x84(%ebp),%eax
804892c:    83 ec 0c            sub    $0xc,%esp
804892f:    50                  push   %eax
8048930:    e8 ea fe ff ff      call   804881f <error>
8048935:    83 c4 10            add    $0x10,%esp
8048938:    8b 95 7c ff ff ff   mov    -0x84(%ebp),%edx
804893e:    a1 48 a0 04 08      mov    0x804a048,%eax
8048943:    39 c2               cmp    %eax,%edx
8048945:    74 05               je     804894c <main+0x106>
8048947:    e8 f4 fe ff ff      call   8048840 <Bomba>
804894c:    83 ec 08            sub    $0x8,%esp
804894f:    6a 00               push   $0x0
8048951:    8d 45 80            lea    -0x80(%ebp),%eax
8048954:    50                  push   %eax
8048955:    e8 46 fb ff ff      call   80484a0 <gettimeofday@plt>
804895a:    83 c4 10            add    $0x10,%esp
804895d:    8b 55 80            mov    -0x80(%ebp),%edx
8048960:    8b 45 88            mov    -0x78(%ebp),%eax
8048963:    29 c2               sub    %eax,%edx
8048965:    89 d0               mov    %edx,%eax
8048967:    83 f8 05            cmp    $0x5,%eax
804896a:    7e 05               jle    8048971 <main+0x12b>
```

En 80491e es donde se llama a la función que pide el código por pantalla. Como sabemos por el apartado de desactivación de la bomba, la función gettimeofday@plt corresponde a la parte de la activación de la bomba cuando pasa un periodo de tiempo, por lo tanto el código que nos queda por revisar es:

```
8048923:    83 c4 10            add    $0x10,%esp
8048926:    8b 85 7c ff ff ff   mov    -0x84(%ebp),%eax
804892c:    83 ec 0c            sub    $0xc,%esp
804892f:    50                  push   %eax
8048930:    e8 ea fe ff ff      call   804881f <error>
8048935:    83 c4 10            add    $0x10,%esp
8048938:    8b 95 7c ff ff ff   mov    -0x84(%ebp),%edx
804893e:    a1 48 a0 04 08      mov    0x804a048,%eax
8048943:    39 c2               cmp    %eax,%edx
8048945:    74 05               je     804894c <main+0x106>
8048947:    e8 f4 fe ff ff      call   8048840 <Bomba>
804894c:    83 ec 08            sub    $0x8,%esp
804894f:    6a 00               push   $0x0
8048951:    8d 45 80            lea    -0x80(%ebp),%eax
8048954:    50                  push   %eax
```

Y aquí observamos que hay dos funciones sospechosas: <error> y <Bomba>

Ejecutamos pasa a paso para entrar en la función error y ver que ocurre ahí (con stepi llegado a esa línea:

```
(gdb) stepi
0x0804881f in error ()
(gdb)
```

Miramos en el código ensamblador:

```
0804881f <error>:
 804881f:       55                      push   %ebp
 8048820:       89 e5                   mov    %esp,%ebp
 8048822:       83 ec 18                sub    $0x18,%esp
 8048825:       a1 48 a0 04 08          mov    0x804a048,%eax
 804882a:       83 e8 80                sub    $0xffffff80,%eax
 804882d:       89 45 f4                mov    %eax,-0xc(%ebp)
 8048830:       8b 45 08                mov    0x8(%ebp),%eax
 8048833:       3b 45 f4                cmp    -0xc(%ebp),%eax
 8048836:       74 05                   je     804883d <error+0x1e>
 8048838:       e8 ee fd ff ff          call   804862b <boom>
 804883d:       90                      nop
 804883e:       c9                      leave
 804883f:       c3                      ret

08048840 <Bomba>:
 8048840:       55                      push   %ebp
 8048841:       89 e5                   mov    %esp,%ebp
 8048843:       90                      nop
 8048844:       5d                      pop    %ebp
 8048845:       c3                      ret
```

De aquí se puede ver a simple vista que la función Bomba no hace nada, ya que tal como entra hace una operación del tipo "nop", y se sale sin realizar nada. Sin embargo en la función error están ocurriendo mas operaciones.
Vemos que se pasa a %eax el valor de 0x804a048, compruebo que valor contiene esa posición el registro eax:

```
(gdb) nexti
0x08048830 in error ()
(gdb) print $eax
$5 = 7782
```

Obtengo el valor 7782, el cual es bastante sospechoso. Ejecuto el programa para ver si ese código es correcto:

```
tehribbon@tehribbon-SATELLITE-L50-B:~/Escritorio/INFORMATICA/curso_recu/1°Cuatri
mestre/EC/PRACTICAS/sesion4/Ficheros_fuente$ ./bomba_Mario_Lopez_Ruiz
Introduce la contraseña: desactivada
Introduce el código: 7782
************************
*** bomba desactivada ***
************************
```

Y efectivamente, he encontrado también el valor del código.