UNIVERSIDAD
DE GRANADA

Estructura de Computadores
Grado en Ingeniería Informática
Curso 2018-2019

Memoria Práctica 4.
Bomba Digital - Desensambladores.

Félix Ramírez García
felixramirezgarcia@correo.ugr.es

14 de noviembre de 2018

Índice

Índice de figuras

Índice de tablas

1. 4.1-Programar la bomba digital

A continuación se va a explicar como desactivar mi bomba paso a paso, para ello abrimos el ejecutable con gdb , usamos layout asm y layout regs para activar el entorno gráfico , ponemos un breakpoint en main y ejecutamos.

Vamos avanzando con nexti hasta que nos solicite la contraseña, le introducimos una al azar y seguimos avanzando hasta que nos encontramos con la llamada a la función password_chars . Usamos el comando stepi para entrar a la función cuyo código es el que muestra la figura 1.1 .

```
000000000040071b <password_chars>:
  40071b: 0f b6 57 01           movzbl 0x1(%rdi),%edx
  40071f: 80 fa 65              cmp    $0x65,%dl
  400722: 75 76                 jne    40079a <password_chars+0x7f>
  400724: 0f b6 47 07           movzbl 0x7(%rdi),%eax
  400728: 0f b6 4f 04           movzbl 0x4(%rdi),%ecx
  40072c: 88 4f 07              mov    %cl,0x7(%rdi)
  40072f: 88 47 04              mov    %al,0x4(%rdi)
  400732: b8 00 00 00 00        mov    $0x0,%eax
  400737: 0f b6 4f 03           movzbl 0x3(%rdi),%ecx
  40073b: 80 f9 6f              cmp    $0x6f,%cl
  40073e: 75 61                 jne    4007a1 <password_chars+0x86>
  400740: 0f b6 77 08           movzbl 0x8(%rdi),%esi
  400744: 88 57 08              mov    %dl,0x8(%rdi)
  400747: 40 88 77 01           mov    %sil,0x1(%rdi)
  40074b: 0f b6 57 07           movzbl 0x7(%rdi),%edx
  40074f: 80 fa 6c              cmp    $0x6c,%dl
  400752: 75 52                 jne    4007a6 <password_chars+0x8b>
  400754: 0f b6 77 04           movzbl 0x4(%rdi),%esi
  400758: 44 0f b6 07           movzbl (%rdi),%r8d
  40075c: 44 88 47 04           mov    %r8b,0x4(%rdi)
  400760: 40 88 37              mov    %sil,(%rdi)
  400763: 80 3f 69              cmpb   $0x69,(%rdi)
  400766: 75 2e                 jne    400796 <password_chars+0x7b>
  400768: 80 7f 01 6d           cmpb   $0x6d,0x1(%rdi)
  40076c: 75 28                 jne    400796 <password_chars+0x7b>
  40076e: 80 7f 02 70           cmpb   $0x70,0x2(%rdi)
  400772: 75 22                 jne    400796 <password_chars+0x7b>
  400774: 80 f9 6f              cmp    $0x6f,%cl
  400777: 75 1d                 jne    400796 <password_chars+0x7b>
  400779: 80 7f 04 73           cmpb   $0x73,0x4(%rdi)
  40077d: 75 17                 jne    400796 <password_chars+0x7b>
  40077f: 80 7f 05 69           cmpb   $0x69,0x5(%rdi)
  400783: 75 11                 jne    400796 <password_chars+0x7b>
  400785: 80 7f 06 62           cmpb   $0x62,0x6(%rdi)
  400789: 75 0b                 jne    400796 <password_chars+0x7b>
  40078b: 80 fa 6c              cmp    $0x6c,%dl
  40078e: 75 06                 jne    400796 <password_chars+0x7b>
  400790: 80 7f 08 65           cmpb   $0x65,0x8(%rdi)
  400794: 74 03                 je     400799 <password_chars+0x7e>
  400796: 83 c0 01              add    $0x1,%eax
  400799: c3                    retq
```

Figura 1.1: .

Mientras seguimos avanzando nos damos cuenta de que la palabra que hemos introducido se guarda en rdi ,y lo primero que hace es comparar la primera posición de la palabra con la letra 'e' , y si esto es cierto realiza un cambio entre las posiciones 7 y 4 de la palabra

. En caso de que no se cumpla la condición de que la primera posición sea una 'e' se realiza un salto a password_chars+127 , instrucción que pone un uno en eax , condición para que explote la bomba. El código en ensamblador de esta parte de la función es que muestra la figura 1.2 .

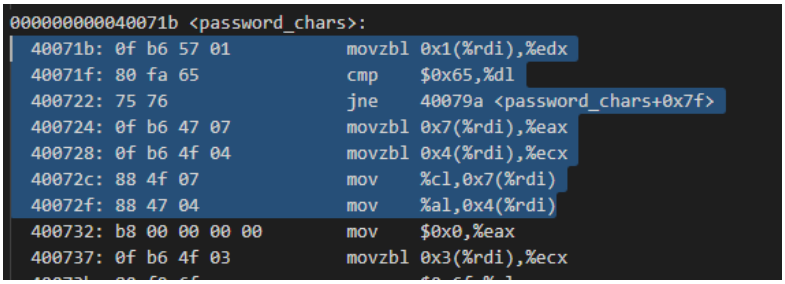

Figura 1.2: Primer swap de chars de la funcion password char..

Mientras seguimos avanzando nos damos cuenta de que esta misma estructura se repite dos veces mas , una segunda comprobando si la tercera posición de la palabra es la letra 'o' , y si lo es, cambia las letras de las posiciones 8 y 1 . En caso de que no se cumpla la condición de que la tercera posición sea una 'o' se realiza un salto a password_chars+134 , instrucción que aumenta en uno el valor de eax , condición para que explote la bomba. El código de esta segunda parte es el muestra la figura 1.3

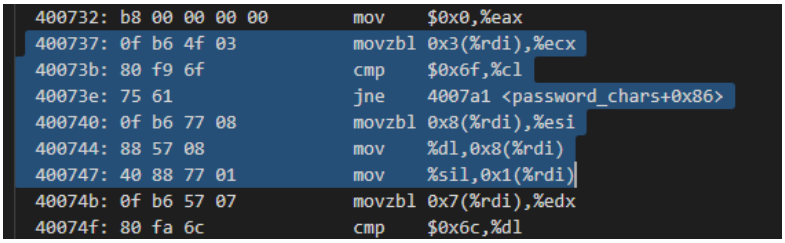

Figura 1.3: Segundo swap de chars de la funcion password char..

En la tercera vez que se repite esta estructura se compara la séptima posición de la palabra con la letra 'l' , y en caso de que lo sea se realiza un cambio de letras entre las posiciones 4 y 0 de la palabra. En caso de que no se cumpla la condición de que la séptima posición sea una 'l' se realiza un salto a password_chars+139 , instrucción que aumenta en uno el valor de eax , condición para que explote la bomba. El código de esta tercera parte es el que muestra la figura 1.4.

Por lo tanto ya tenemos que la primera posición (Empezando a contar por el 0) de la palabra tiene la letra 'e' , la tercera posición de la palabra tiene la 'o', y la séptima posición tiene la 'l' . Para pasar esta fase hay que introducir una palabra con la primera letra una 'e' , la tercera letra una 'o' , y en la cuarta letra una 'l' (Porque se cambia en el primer swap con la posición 7).

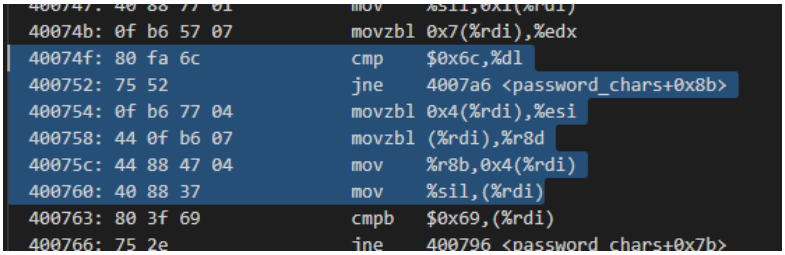

Figura 1.4: Tercer swap de chars de la funcion password char.

Al seguir avanzando , nos encontramos con nueve comprobaciones mas , las que muestra la figura 1.5 .

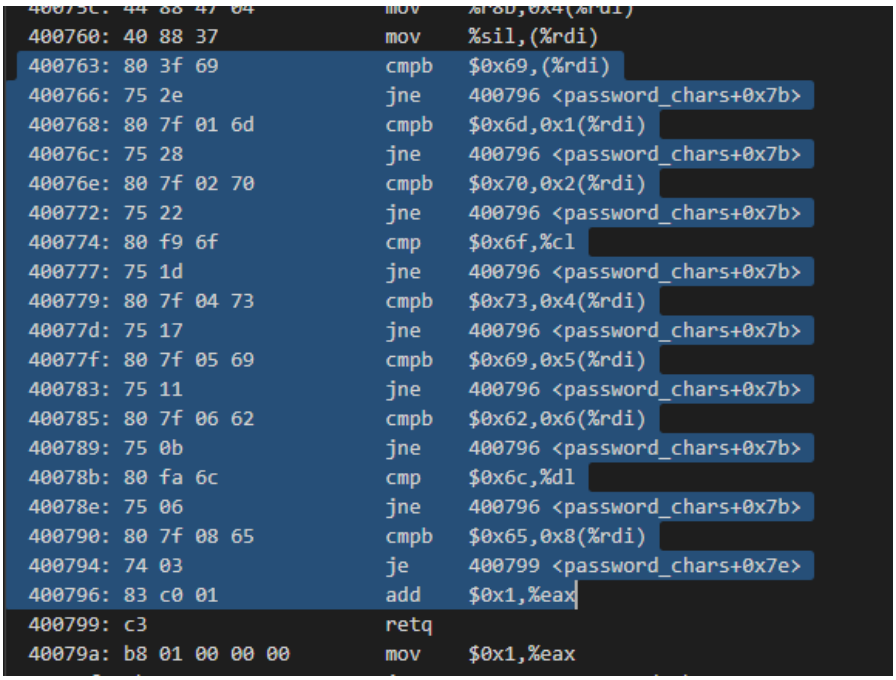

Figura 1.5: Comprobaciones finales de password char.

Si cualquiera de estas nueve comparaciones falla, se saltaría a password_chars+126 , instrucción que aumenta en uno el valor de eax , condición para que explote la bomba.

La primera condición comprueba si la letra en la posición numero cero es la 'i' (0x69), la segunda condición comprueba si la letra en la posición numero uno es la 'm' (0x6d), la tercera condición comprueba si la letra en la posición numero dos es la 'p' (0x70), la cuarta condición comprueba si la letra en la posición numero tres es la 'o' (0x6f), la quinta condición comprueba si la letra en la posición numero cuatro es la 's' (0x73),

la sexta condición comprueba si la letra en la posición numero cinco es la 'i' (0x69), la séptima condición comprueba si la letra en la posición numero seis es la 'b' (0x62), la octava condición comprueba si la letra en la posición numero siete es la 'l' (0x6c) y la novena condición comprueba si la letra en la posición numero ocho es la 'e' (0x65) .

A partir de aquí se puede deducir que la palabra encriptada es 'imposible' y ahora tenemos que recorrer de forma inversa la función para simular el proceso de encriptacion del final a principio. Por lo tanto tenemos que recorrer los tres condicionales comentados anteriormente para saber cual es la palabra sin encriptar.

Partiendo ahora de la palabra 'imposible' , vamos a aplicarle el tercer condicional , que cambia las posiciones 0 y 4 . Ahora tenemos la palabra 'smpoiible' . Después le aplicamos el segundo condicional , que cambia las letras de las posiciones 8 y 1 , quedándose la palabra en 'sepoiiblm' . Por último le aplicamos el cambio que realiza el primer condicional que nos encontramos al entrar en la función , el que intercambia las posiciones 7 y 4, quedándose la palabra resultado en 'sepolibim'. Para una comprensión mas clara de esta ultima parte esta la figura 1.6 .

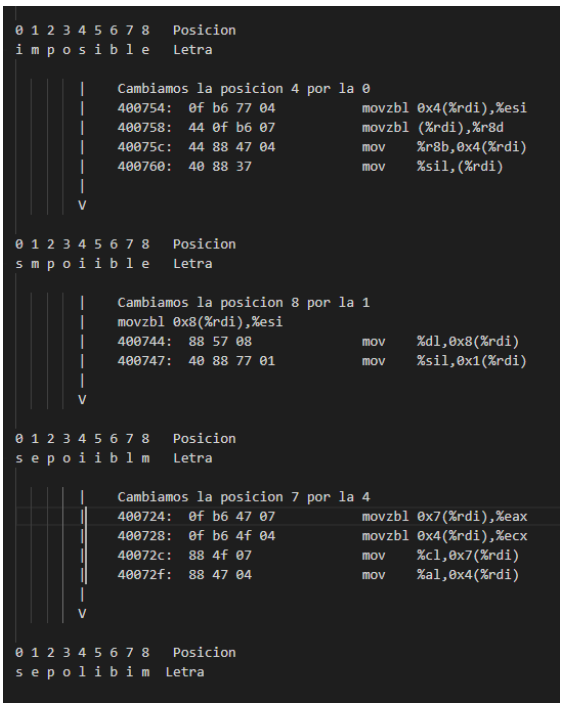

Figura 1.6: Etapas de desencriptado.

Una vez tenemos la palabra y no nos explota la bomba en esta función podemos seguir avanzando hasta encontrarnos con la llamada a la segunda función , password_number .Introducimos algo al azar y una vez dentro con stepi podemos ver en su código que

almacena el numero en el registro edi, el código es el que muestra la figura 1.7 .

```
00000000004007ab <password_number>:
  4007ab:       83 ff 6f                cmp    $0x6f,%edi
  4007ae:       75 06                   jne    4007b6 <password_number+0xb>
  4007b0:       b8 00 00 00 00          mov    $0x0,%eax
  4007b5:       c3                      retq
  4007b6:       b8 01 00 00 00          mov    $0x1,%eax
  4007bb:       c3                      retq
```

Figura 1.7: Funcion password number.

Aquí simplemente se hace una comprobación con 0x6f , y si es correcta , la función devuelve un 0 , condición para desactivar la bomba. Por lo tanto esta contraseña no es mas que el numero 0x6f en decimal . El numero 111 .

Una vez sabemos todo esto se puede ejecutar el programa con las contraseñas 'sepolibim' y '111' y desactivar la bomba . Ahora vamos a modificar ambas contraseñas del ejecutable , para ello vamos a sustituir la letra p de la contraseña 'sepolivim' por una v . Esto lo hacemos abriendo el ejecutable con ghex y buscando la secuencia en hexadecimal que realiza la función password_chars para la comprobación de la palabra almacenada en la posición numero dos (Empezando por la cero) de rdi .La figura 1.8 muestra la búsqueda de la secuencia en hexadecimal '80 7f 02 70'.

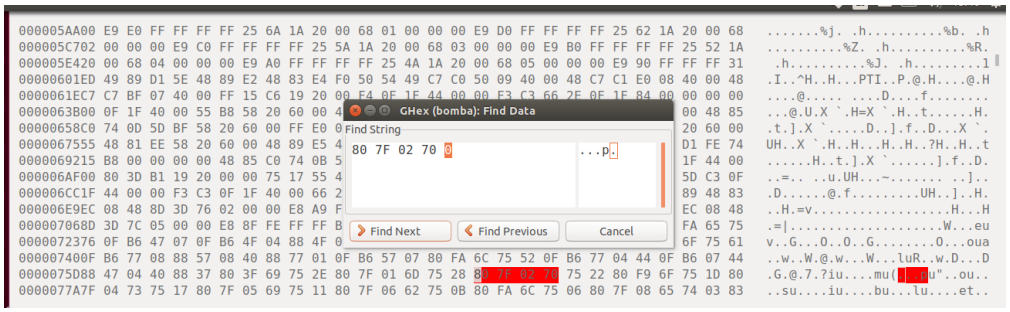

Figura 1.8: Busqueda de la comprobacion de la palabra para la letra p y sustitucion por la v

Como se puede apreciar en la figura ,el par de dígitos que representa a la letra p es el '70' , por lo que si los sustituimos por 76 habremos cambiado la palabra 'sepolibim' por 'sevolibim'.

Por último vamos a modificar la contraseña numérica , para ello buscamos en ghex la secuencia en hexadecimal que hace la comprobación con el numero 11 , la secuencia '83 ff 6f' , y modificamos el valor de 6f por el de 7f , cambio que muestra la figura 1.9 . Quedando ahora la contraseña numérica en 127.
Para concluir esta sección se van a mostrar en la figura 1.10 la ejecución de ambas bombas , una primera sin modificaciones , y una segunda alterada .

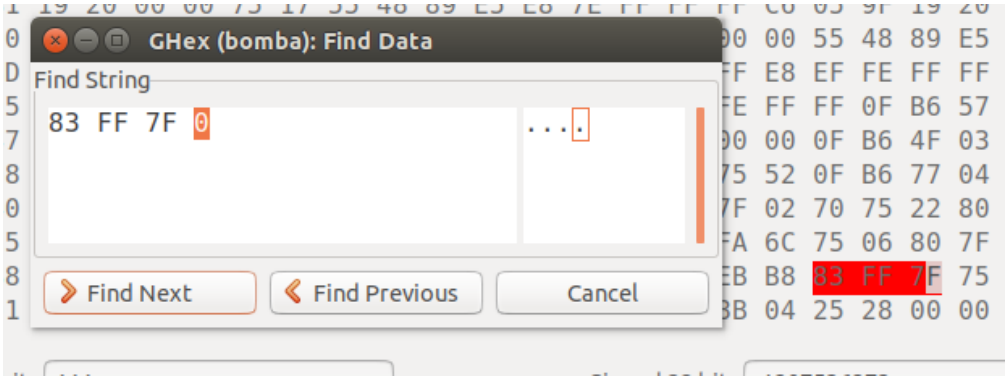

Figura 1.9: Busqueda de la comprobacion del numero 111 y sustitucion por el 127

```
felix@felix-VirtualBox:~/Escritorio/Practica4/Codigo$ ./bomba

Introduce la contraseña: sepolibim

Introduce el pin: 111

.......................
··· bomba desactivada ···
.......................

felix@felix-VirtualBox:~/Escritorio/Practica4/Codigo$ ./bomba_alterada

Introduce la contraseña: sevolibim

Introduce el pin: 127

.......................
··· bomba desactivada ···
.......................

felix@felix-VirtualBox:~/Escritorio/Practica4/Codigo$
```

Figura 1.10: Ejecucion de ambas bombas