Explicación de la Bomba

Por: Arturo Cortés Sánchez

contraseña: secuRepass
pin: -788

contraseña modificada: secuLepass
pin modificado: -782

Explicación:

Contraseña:

Abrimos la bomba con gdb y avanzamos con “ni”. Introducimos una contraseña cualquiera y seguimos avanzando hasta que veamos la llamada a la función “cifradosuperseguro”.

```
> 0x4011ba <cifradosuperseguro>          mov    $0x11,%eax
  0x4011bf <cifradosuperseguro+5>        jmp    0x4011d5 <cifradosuperseguro+27>
  0x4011c1 <cifradosuperseguro+7>        movslq %eax,%rdx
  0x4011c4 <cifradosuperseguro+10>       lea    0x2eb5(%rip),%rcx        # 0x404080 <password>
  0x4011cb <cifradosuperseguro+17>       movzbl (%rcx,%rdx,1),%ecx
  0x4011cf <cifradosuperseguro+21>       xor    %cl,(%rdi,%rdx,1)
  0x4011d2 <cifradosuperseguro+24>       sub    $0x1,%eax
  0x4011d5 <cifradosuperseguro+27>       test   %eax,%eax
  0x4011d7 <cifradosuperseguro+29>       jg     0x4011c1 <cifradosuperseguro+7>
  0x4011d9 <cifradosuperseguro+31>       retq
```

Lo primero que hace la función es guardar 0x11 en %eax, luego da un salto hacia abajo donde comprueba si %eax es mayor que 0. En caso positivo vuelve a saltar, esta vez hacia arriba, justo debajo del primer salto. Viendo esto podemos empezar a intuir un bucle for que empieza en 17 (0x11) y acaba en 0.

Siguiendo con la función vemos como guarda el valor de %eax en %rdx y luego guarda la dirección de “password” en %rcx. Si comprobamos la dirección de “password” vemos que contiene el string “contrasena_segura”. Posteriormente accede a %rcx+%rdx y lo guarda en %ecx. Esto vendría a ser lo mismo que guardar en %ecx el contenido de “password+i”

(gdb) x/1bs 0x404080
0x404080 <password>: "contrasena_segura"

En la siguiente instrucción podemos ver que se hace uso de %rdi. Si comprobamos la dirección de memoria que contiene %rdi vemos que guarda la contraseña que hemos introducido. Dicha instrucción hace un xor entre %cl (recordemos que contiene “password+i”) y %rdi+%rdx, o lo que es equivalente “contraseña introducida + i”.

```
┌─Register group: general──────────────────────────────────────────────────────┐
│rax            0x11                17                                          │
│rbx            0x7fffffffe170      140737488347504                             │
│rcx            0x0                 0                                           │
│rdx            0x11                17                                          │
│rsi            0x7ffff7f73730      140737353561904                             │
│rdi            0x7fffffffe170      140737488347504                             │
│rbp            0x401330            0x401330 <__libc_csu_init>                  │
│rsp            0x7fffffffe138      0x7fffffffe138                              │
│r8             0x40567b            4216443                                     │
│r9             0x7ffff7f73720      140737353561888                             │
│r10            0x7ffff7d8bb80      140737351564160                             │
│r11            0x246               582                                         │
│r12            0x4010a0            4198560                                     │
│r13            0x7fffffffe2c0      140737488347840                             │
│r14            0x0                 0                                           │
│r15            0x0                 0                                           │
└──────────────────────────────────────────────────────────────────────────────┘
```

(gdb) x/1bs 0x7fffffffe170
0x7fffffffe170: "secuRepass\n"

Si avanzamos varias iteraciones en el bucle y volvemos a comprobar la dirección de la contraseña, vemos que esta ha sido modificada.

Una vez acabado el bucle el programa sale de la función y vuelve al main donde vemos que el programa guarda en %rsi la direccion de algo llamado “p” y en %rdi la contraseña modificada que hemos introducido. Luego llama a “strncmp”. Viendo esto podemos suponer que “p” contiene la contraseña cifrada, si lo comprobamos obtenemos un string de 10 caracteres, en su mayoría no imprimibles.

```
   0x401243 <main+71>      mov    $0x64,%esi
   0x401248 <main+76>      callq  0x401070 <fgets@plt>
   0x40124d <main+81>      test   %rax,%rax
   0x401250 <main+84>      je     0x401226 <main+42>
   0x401252 <main+86>      lea    0x30(%rsp),%rbx
   0x401257 <main+91>      mov    %rbx,%rdi
   0x40125a <main+94>      callq  0x4011ba <cifradosuperseguro>
   0x40125f <main+99>      mov    $0xa,%edx
 > 0x401264 <main+104>     lea    0x2dfd(%rip),%rsi        # 0x404068 <p>
   0x40126b <main+111>     mov    %rbx,%rdi
   0x40126e <main+114>     callq  0x401030 <strncmp@plt>
   0x401273 <main+119>     test   %eax,%eax
   0x401275 <main+121>     je     0x40127c <main+128>
   0x401277 <main+123>     callq  0x401186 <boom>
   0x40127c <main+128>     lea    0x20(%rsp),%rdi
   0x401281 <main+133>     mov    $0x0,%esi
```

(gdb) x/1bs 0x404068
0x404068 <p>: "s\n\r\001 \004\003\004\035\022"

Sabiendo que al aplicar una operación xor dos veces sobre sobre un mismo carácter obtenemos el carácter original, podemos suponer que si aplicamos el mismo algoritmo de cifrado al string que acabamos de obtener, encontraremos la contraseña.

Primero necesitamos encontrar el valor hexadecimal de dichos caracteres:

(gdb) x/10bx 0x404068
0x404068 <p>: 0x73 0x0a 0x0d 0x01 0x20 0x04 0x03 0x04
0x404070 <p+8>: 0x1d 0x12

Con dichos valores y lo que sabemos, podemos intentar reproducir la función de cifrado (que es la misma que la de descifrado):
Un bucle for que recorre un string formado por los caracteres no imprimibles de forma descendente. A cada carácter i del string se le aplica una operación xor con el carácter i del string "contrasena_segura".

```c
#include <stdio.h>

char a[] = "contrasena_segura";

void descifradosuperseguro(char *input, int size) {
  for (int i = size; i > 0; i--) {
    input[i] = input[i] ^ a[i];
  }
}

int main() {
  char p[] = "\x73\xa\xd\x1\x20\x4\x3\x4\x1d\x12";
  descifradosuperseguro(p, sizeof(p) - 2); /*-1 por el \n y -1 por el fin de string*/
  printf("%s\n", p);
}
```

Y así obtenemos la contraseña: secuRepass

Pin:

Una obtenida la contraseña, volvemos a ejecutar el programa avanzamos hasta fgets, la introducimos y avanzamos un poco. Si todo ha ido bien la bomba no debería explotar. Seguimos avanzando hasta la comprobación del tiempo. Para que la bomba no explote ahí ponemos %rax a 0 con "set $rax=0x0". Seguimos avanzando hasta que el programa nos pida el pin, introducimos cualquier numero y continuamos hasta la la llamada a la función "cifradoultraseguro".

```
> 0x4011da <cifradoultraseguro>          mov    %edi,%eax
  0x4011dc <cifradoultraseguro+2>        mov    $0x0,%edx
  0x4011e1 <cifradoultraseguro+7>        cmp    $0x11,%edx
  0x4011e4 <cifradoultraseguro+10>       ja     0x4011fb <cifradoultraseguro+33>
  0x4011e6 <cifradoultraseguro+12>       movslq %edx,%rcx
  0x4011e9 <cifradoultraseguro+15>       lea    0x2e90(%rip),%rsi        # 0x404080 <password>
  0x4011f0 <cifradoultraseguro+22>       movsbl (%rsi,%rcx,1),%ecx
  0x4011f4 <cifradoultraseguro+26>       add    %ecx,%eax
  0x4011f6 <cifradoultraseguro+28>       add    $0x1,%edx
  0x4011f9 <cifradoultraseguro+31>       jmp    0x4011e1 <cifradoultraseguro+7>
  0x4011fb <cifradoultraseguro+33>       retq
```

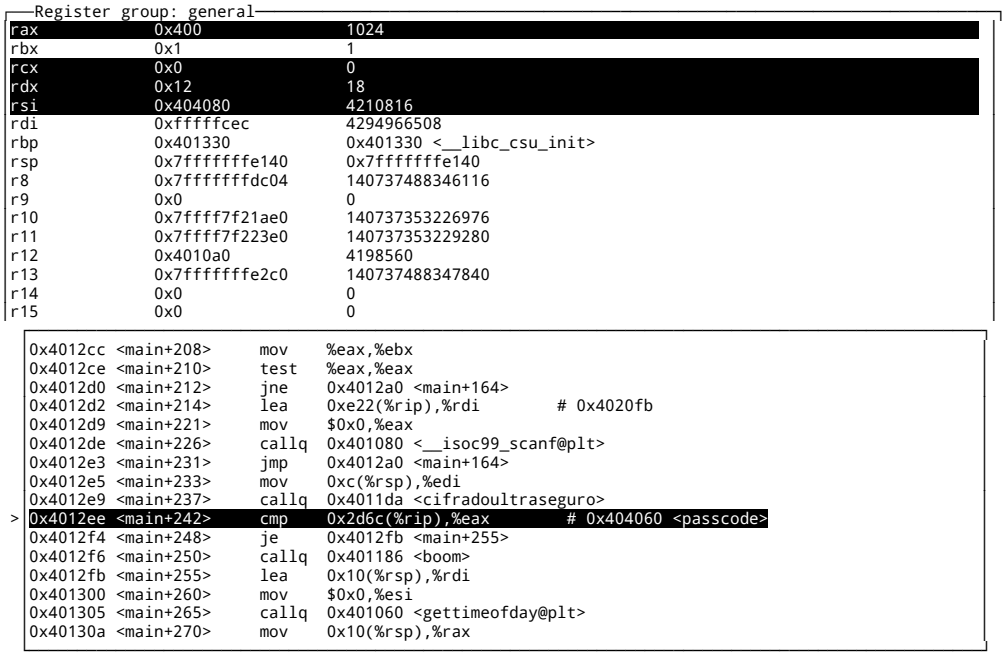
En "cifradoultraseguro" vemos que guarda en %eax el numero que hemos introducido, pone a 0 %edx y lo compara con 0x11 (17). Si es mayor salta al final de la función. Aquí podemos empezar a intuir un bucle for ascendente que acaba en 17 donde %eax es i. A continuación guarda el valor de %edx en %rcx y la dirección de "password" en %rsi. En las siguientes instrucciones vemos como guarda el contenido de la dirección equivalente a %rsi+%rcx (o lo que es lo mismo "password+i") en %ecx y lo suma a %eax (pin introducido), luego suma 1 a %edx y vuelve a empezar.

Si avanzamos un poco en el bucle podemos ver que va sumando al pin introducido el valor numérico de cada carácter del string “contrasena_segura”.

Avanzamos hasta salir de la función y vemos como se compara una dirección que contiene el codigo secreto (“passcode”) con %rax. Miramos el valor de “passcode”:

(gdb) x/1wd 0x404060
0x404060 <passcode>: 1024

%rax contiene el pin que hemos mas el valor numérico de cada carácter de “password”. Por lo tanto si a %rax le restamos el numero que hemos introducido obtenemos lo que “cifradoultraseguro” le ha sumado al pin, 1812. Por tanto necesitamos un número que sumado a 1812 de 1024.
Ese número es -788



Cambiando la contraseña y el pin

Ya que el string “contrasena_segura” es usado en el calculo de ambas funciones, modificar el string hace que se modifiquen tanto la contaseña como el pin.

Abrimos el ejecutable con ghex, buscamos el string y cambiamos la primera r por una l. Nos queda “contlasena_segura”. He decidido cambiar dicha letra ya que en el cifrado se le hace xor con 0x20 y una letra minúscula xor 0x20 da como resultado dicha letra en mayúscula. Por lo que la contraseña modificada sería “secuLepass”

Para el pin lo tenemos mas fácil, solo tenemos que cambiar en la suma realizada en “cifradoultraseguro” el valor del carácter r por el valor del carácter r. Para ello miramos una tabla ASCII y buscamos el valor decimal de l y r, que son 108 y 114 respectivamente. Por tanto para obtener el nuevo pin tenemos que hacer la siguiente operación: 1024-(1812-114+108), lo cual da -782 que es el nuevo pin.