Defusando la Bomba

Subsecciones:

1- Depuración Automatizada.
2 - Depuración en Profundidad.
3 - Descifrando el Cifrado.

Soluciones:

Original: **LinuxRocks**
Encriptada: **GcgmoHdW^e**
Original: **69420**
Encriptada= **71440**

1- Depuración Automatizada.

Disponemos de un explain.gdb para adquirir los valores de las contraseñas de forma automática y además de así ir analizando la depuración del ejecutable.

El archivo explain.gdb debe ser ejecutado en la misma carpeta del ejecutable con el siguiente comando de la terminal: gdb -q -x explain.gdb; Y solamente tendremos que darle a “ENTER” un par de veces para defusar la bomba.

2 - Depuración en Profundidad.

Primeramente debemos inicializar gdb, y además, las pantallas de depuración que nos ayudarán a guiarnos a tráves del código: gdb -tui ./bomba;

Reading symbols from bomba...(no debugging symbols found)...done.
(gdb) layout asm
(gdb) layout regs
(gdb) br main
Breakpoint 1 at 0x40077d
(gdb) run < <(echo -e hacking\\n1234\\n)
Starting program: *{CENSORED }*/PRACTICAS/EC/P4/GDBfiles/explain/bomba < <(echo -e hacking\\n1234\\n)

Breakpoint 1, 0x000000000040077d in main ()

Ahora tenemos que localizar dónde se realizan las comprobaciones de password and passcode en el ejecutable. Para ello disponemos de una ayuda que sale en la pantalla asm, **layout asm**, así que nos centraremos en ella y buscaremos **“ # 0x601068 password “** para ello tenemos que ejecutar **focus asm** e ir bajando hasta encontrar el siguiente **break** que tengamos que realizar.

```
0x4007e0 <main+89>      je     0x4007b1 <main+42>
0x4007e2 <main+91>      lea    0x30(%rsp),%rbx
```

```
0x4007e7 <main+96>      mov    %rbx,%rdi
0x4007ea <main+99>      callq  0x400727 <encriptar>
0x4007ef <main+104>     mov    $0xc,%edx
0x4007f4 <main+109>     lea    0x20086d(%rip),%rsi  #0x601068<password>
0x4007fb <main+116>     mov    %rbx,%rdi
0x4007fe <main+119>     callq  0x4005d0 <strncmp@plt>
0x400803 <main+124>     test   %eax,%eax
0x400805 <main+126>     je     0x40080c <main+133>
```

Por lo que ahora sabemos que tenemos que ejecutar **br *main+109** y **continue** para posicionarnos donde podemos acceder a la variable password. Para ello tecleamos: **p (char[0xd])password;** y ahora tenemos la contraseña **encriptada**. Antes de decifrar-la, creo oportuno encontrar ya el **pin** para ello tenemos que saltarnos 2 comprobaciones y seguir adelante. Iremos paso a paso:

1. Saltar primer test, el de la contraseña: vemos que el "**test**" se ejecutaba en ***main+124** Y estamos en ***main+103** así que con un par de **ni** estamos donde deseamos.
2. Una vez en el test, queremos que se cumpla la condición así que haremos un **set $eax=0,** podemos hacer **layout regs** para comprobarlo.
3. Ahora tenemos que hacer un br ***main+158** seguido de un **continue** para poder falsificar la prueba de tiempo con el mismo comando **set $eax=0**.

Ahora buscamos **#0x601060 passcode:**

```
0x40086e <main+231>     callq  0x400620 <__isoc99_scanf@plt>
0x400873 <main+236>     cmp    $0x1,%ebx
0x400876 <main+239>     jne    0x400830 <main+169>
0x400878 <main+241>     lea    0xc(%rsp),%rdi
0x40087d <main+246>     callq  0x40074c <encriptarp>
0x400882 <main+251>     mov    0x2007d8(%rip),%eax #0x601060<passcode>
0x400888 <main+257>     cmp    %eax,0xc(%rsp)
0x40088c <main+261>     je     0x400893 <main+268>
```

Hacemos un **br *main+251** seguido de un **continue** y ahora: **p (int)passcode**.

3. Descifrando el Cifrado.

Seguindo los pasos de la subsección anterior nos hemos quedado con la siguiente información:

Contraseña Encriptada: GcgmoHdW^e

Pin Encriptado = 77756;

Y además hemos podido notar que hay una llamada a una función encriptar poco antes de la comprobación entre los strings. Vamos a analizar-la:

```
    0x400727 <encriptar>    mov    $0x0,%edx  //Inicializamos la variable
    0x40072c <encriptar+5>  cmp    $0x63,%edx //Comparamos con SIZE
```

```
0x40072f <encriptar+8>  jg     0x40073e <encriptar+23>
0x400731 <encriptar+10> movslq %edx,%rcx //Creamos copia del contador
0x400734 <encriptar+13> add    %rdi,%rcx //Array[i];
0x400737 <encriptar+16> movzbl (%rcx),%eax //Creamos copia;
0x40073a <encriptar+19> cmp    $0xa,%al //Comprobar valor válido.
0x40073c <encriptar+21> jne    0x400740 <encriptar+25>
0x40073e <encriptar+23> repz retq
0x400740 <encriptar+25> sub    %edx,%eax // A[i]-=i;
0x400742 <encriptar+27> sub    $0x5,%eax // A[i]-=5;
0x400745 <encriptar+30> mov    %al,(%rcx)
0x400747 <encriptar+32> add    $0x1,%edx  //Incrementar contador
0x40074a <encriptar+35> jmp    0x40072c <encriptar+5>
0x40074c <encriptarp>   addl   $0x7e4,(%rdi) //Suma fija de valor;
0x400752 <encriptarp+6> retq
```

Como podemos observar leyendo los comentarios, el código aunque parezca difícil es un simple desplazamiento dependiente de la posición del carácter en la cadena y además de uno fijo de valor igual a 5.

En código C, sería algo similar a:

```
for(int i=0;password[i]!='\0';i++){
      password[i]=password[i]-i-5;
}
```

Por los que tendríamos que ir sumando 5,6,7,8,9,10… y así sobre la contraseña encriptada que hemos obtenido para entonces hallar la **original: LinuxRocks**

Para obtener el pin podemos mirar la llamada a **encriptarp** que es una simple suma de un valor fijo: **0x7e4 -> 2020** tal que la original es:

Original = 69420