Bomba: bombaProf5_15

Resuelta por Juan Sánchez Rodríguez

Para resolverla he usado la herramienta gdb. Empezamos usando la instrucción “gdb bomba_prof5_15”, después, dentro del gdb usamos la instrucción “break main” para colocar un breakpoint en el main, usamos “run” y después “disas” para ver el programa:

```
0x0804865c <+0>:  push   %ebp
0x0804865d <+1>:  mov    %esp,%ebp
0x0804865f <+3>:  and    $0xfffffff0,%esp
0x08048662 <+6>:  push   %edi
0x08048663 <+7>:  push   %ebx
0x08048664 <+8>:  sub    $0x98,%esp
0x0804866a <+14>:         mov    %gs:0x14,%eax
0x08048670 <+20>:         mov    %eax,0x8c(%esp)
0x08048677 <+27>:         xor    %eax,%eax
0x08048679 <+29>:         movl   $0x0,0x4(%esp)
0x08048681 <+37>:         lea    0x1c(%esp),%eax
0x08048685 <+41>:         mov    %eax,(%esp)
0x08048688 <+44>:         call   0x80484ac <gettimeofday@plt>
0x0804868d <+49>:         movl   $0x80488ea,0x4(%esp)
0x08048695 <+57>:         movl   $0x1,(%esp)
0x0804869c <+64>:         call   0x804847c <__printf_chk@plt>
0x080486a1 <+69>:         mov    0x804a3a0,%eax
0x080486a6 <+74>:         mov    %eax,0x8(%esp)
0x080486aa <+78>:movl   $0x64,0x4(%esp)
0x080486b2 <+86>:         lea    0x28(%esp),%ebx
0x080486b6 <+90>:         mov    %ebx,(%esp)
0x080486b9 <+93>:         call   0x804848c <fgets@plt>
0x080486be <+98>:         mov    $0x804a260,%edi
0x080486c3 <+103>:        mov    $0x0,%eax
0x080486c8 <+108>:        mov    $0xffffffff,%ecx
0x080486cd <+113>:        repnz scas %es:(%edi),%al
0x080486cf <+115>:        not    %ecx
0x080486d1 <+117>:        sub    $0x1,%ecx
0x080486d4 <+120>:        mov    %ecx,0x8(%esp)
0x080486d8 <+124>:        movl   $0x804a260,0x4(%esp)
0x080486e0 <+132>:        mov    %ebx,(%esp)
0x080486e3 <+135>:        call   0x80484dc <strncmp@plt>
0x080486e8 <+140>:        test   %eax,%eax
0x080486ea <+142>:        je     0x80486f1 <main+149>
0x080486ec <+144>:        call   0x804860e <boom>
0x080486f1 <+149>:        movl   $0x0,0x4(%esp)
0x080486f9 <+157>:        lea    0x14(%esp),%eax
0x080486fd <+161>:        mov    %eax,(%esp)
0x08048700 <+164>:        call   0x80484ac <gettimeofday@plt>
0x08048705 <+169>:        mov    0x14(%esp),%eax
0x08048709 <+173>:        sub    0x1c(%esp),%eax
0x0804870d <+177>:        cmp    $0x3c,%eax
0x08048710 <+180>:        jle    0x8048717 <main+187>
```

```
0x08048712 <+182>:    call   0x804860e <boom>
0x08048717 <+187>:    movl   $0x8048905,0x4(%esp)
0x0804871f <+195>:    movl   $0x1,(%esp)
0x08048726 <+202>:    call   0x804847c <__printf_chk@plt>
0x0804872b <+207>:    lea    0x24(%esp),%eax
0x0804872f <+211>:    mov    %eax,0x4(%esp)
0x08048733 <+215>:    movl   $0x804891c,(%esp)
0x0804873a <+222>:    call   0x80484cc <__isoc99_scanf@plt>
0x0804873f <+227>:    mov    0x24(%esp),%eax
0x08048743 <+231>:    cmp    0x804a274,%eax
0x08048749 <+237>:    je     0x8048750 <main+244>
0x0804874b <+239>:    call   0x804860e <boom>
0x08048750 <+244>:    movl   $0x0,0x4(%esp)
0x08048758 <+252>:    lea    0x1c(%esp),%eax
0x0804875c <+256>:    mov    %eax,(%esp)
0x0804875f <+259>:    call   0x80484ac <gettimeofday@plt>
0x08048764 <+264>:    mov    0x1c(%esp),%eax
0x08048768 <+268>:    sub    0x14(%esp),%eax
0x0804876c <+272>:    cmp    $0x3c,%eax
0x0804876f <+275>:    jle    0x8048776 <main+282>
0x08048771 <+277>:    call   0x804860e <boom>
0x08048776 <+282>:    call   0x80485c0 <defused>
0x0804877b <+287>:    mov    0x8c(%esp),%edx
0x08048782 <+294>:    xor    %gs:0x14,%edx
0x08048789 <+301>:    je     0x8048795 <main+313>
0x0804878b <+303>:    nop
0x0804878c <+304>:    lea    0x0(%esi,%eiz,1),%esi
0x08048790 <+308>:    call   0x80484bc <__stack_chk_fail@plt>
0x08048795 <+313>:    add    $0x98,%esp
0x0804879b <+319>:    pop    %ebx
0x0804879c <+320>:    pop    %edi
0x0804879d <+321>:    mov    %ebp,%esp
0x0804879f <+323>:    pop    %ebp
0x080487a0 <+324>:    ret
```

Lo primero en lo que me fijo es en "0x080486b9 <+93>: call 0x804848c <fgets@plt>", aquí se que es la función donde introducimos nosotros la contraseña, más adelante veo que hay "0x080486e3 <+135>: call 0x80484dc <strncmp@plt>" y esta función se que compara dos cadenas, nos paramos a ver los "mov" anteriores a esta llamada y vemos uno sospechoso: "0x080486d8 <+124>: movl $0x804a260,0x4(%esp)", usamos la orden "print (char*) 0x804a260" y el gdb nos responde con "$1 = 0x804a260 <password> "lMwYLtpa\n"", probamos introduciendo la clave "lMwYLtpa" y vemos que es correcta.
Para la código numérico nos fijamos en " 0x0804873a <+222>: call 0x80484cc <__isoc99_scanf@plt>", aquí es donde vamos a introducir nuestro entero. Las instrucciones siguientes a esta son:

```
0x0804873f <+227>:    mov    0x24(%esp),%eax
0x08048743 <+231>:    cmp    0x804a274,%eax
0x08048749 <+237>:    je     0x8048750 <main+244>
0x0804874b <+239>:    call   0x804860e <boom>
```

Así que nos fijamos en ese “cmp” y miramos si hay algún número en “0x804a274”, ya que este “cmp” es bastante sospechoso, usando la instrucción “print *(int*) 0x804a274” y el gdb nos responde con “$2 = 3687”, probamos introducir el código “3687” y efectivamente “bomba desactivada”.