CÓMO DESACTIVAR bombaJA
Juan Antonio Villegas Recio

Lo primero ante todo una vez obtenido el ejecutable es hacerle un volcado mediante el comando: objdump -d bombaJA
Así obtendremos el código ensamblador.

0804862b <main>:
...
```
 804866f: e8 ac fd ff ff          call   8048420 <fgets@plt>
 8048674: 83 c4 10                add    $0x10,%esp
 8048677: 83 ec 0c                sub    $0xc,%esp
 804867a: 68 38 a0 04 08          push   $0x804a038
 804867f: e8 dc fd ff ff          call   8048460 <strlen@plt>
 8048684: 83 c4 10                add    $0x10,%esp
 8048687: 83 ec 0c                sub    $0xc,%esp
 804868a: 68 38 a0 04 08          push   $0x804a038
 804868f: e8 cc fd ff ff          call   8048460 <strlen@plt>
 8048694: 83 c4 10                add    $0x10,%esp
 8048697: 83 ec 04                sub    $0x4,%esp
 804869a: 50                      push   %eax
 804869b: 68 38 a0 04 08          push   $0x804a038
 80486a0: 8d 45 94                lea    -0x6c(%ebp),%eax
 80486a3: 50                      push   %eax
 80486a4: e8 e7 fd ff ff          call   8048490 <strncmp@plt>
 80486a9: 83 c4 10                add    $0x10,%esp
 80486ac: 85 c0                   test   %eax,%eax
 80486ae: 74 05                   je    80486b5 <main+0x8a>
 80486b0: e8 f6 fe ff ff          call   80485ab <boom>
 80486b5: 83 ec 0c                sub    $0xc,%esp
```
...

Mirando el código, se puede observar que he llamado tres veces a strncmp(), pero solo una de ellas llama a boom(), aunque las tres tienen los mismos argumentos, y uno de ellos es la longitud de la contraseña, la cual se calcula mediante la función strlen(), llamada con anterioridad, y cuyo argumento está en la dirección apilada justo antes de la llamada a strlen(), y este argumento es la contraseña.
Observando un poco más, cuando se pide el código, justo después se le realiza una serie de operaciones para después compararlo con EDX, comprobemos que el código leído se duplica, se le resta 47 (0x2f) y se le suma 1.
...
```
 8048706: e8 75 fd ff ff          call   8048480
<__isoc99_scanf@plt>
 804870b: 83 c4 10                add    $0x10,%esp
 804870e: 8b 45 90                mov    -0x70(%ebp),%eax
 8048711: 01 c0                   add    %eax,%eax
 8048713: 89 45 90                mov    %eax,-0x70(%ebp)
 8048716: 8b 45 90                mov    -0x70(%ebp),%eax
 8048719: 83 e8 2f                sub    $0x2f,%eax
```

```
 804871c: 89 45 90                  mov    %eax,-0x70(%ebp)
 804871f: 8b 45 90                  mov    -0x70(%ebp),%eax
 8048722: 83 c0 01                  add    $0x1,%eax
 8048725: 89 45 90                  mov    %eax,-0x70(%ebp)
 8048728: 8b 55 90                  mov    -0x70(%ebp),%edx
 804872b: a1 44 a0 04 08            mov    0x804a044,%eax
 8048730: 39 c2                     cmp    %eax,%edx
 8048732: 74 05                     je    8048739 <main+0x10e>
 8048734: e8 72 fe ff ff            call   80485ab <boom>
```

…

Luego EAX almacenará el código que el usuario haya introducido tras las modificaciones, y EDX el auténtico valor del código.

Para resolver la bomba, lo primero que recomiendo es abrir el programa en ghex:
ghex bombaJA &
Una vez abierto, buscar todas las secuencias de instrucciones 74 05 (saltos condicionales JE) y cambiar 74 por EB, para así convertir un salto condicional en uno incondicional JMP, así la bomba se desactivará independientemente del valor que se introduzca en contraseña y código. Repetir este proceso con las secuencias 7E 05 (saltos condicionales JLE) para así despreocuparnos también del tiempo.

En este momento la bomba está desactivada, pero además de todo esto podemos adivinar o incluso cambiar la contraseña y el código, para ello mediante un depurador coloquemos un punto de ruptura (breakpoint) en la llamada a la función strlen() y hagamos correr el programa.
Cuando llegue a ese punto de ruptura, mediante un volcado de memoria (Examine **`1 string bytes`** from **`0x804a038)`** desde la posición indicada anteriormente como argumento de strlen(), nos muestra el programa la contraseña:

`0x804a038 <password>:    “bombafacil\n”`

De igual forma colocando un breakpoint cercano a la comparación y examinando el valor de EDX obtenemos el código: 2564
Pero no es ese el código a introducir, pues el programa lo cambia, el valor a introducir es (2564-1+47)/2=1305.