DESACTIVANDO BOMBA_NBA_15

Daniel Monjas Miguélez

23 de diciembre de 2019

1. Pasos para desactivar la bomba

En primer lugar utilizamos la orden **objdump**, para desemsamblar el código y asi tener una idea de donde se realizan las llamadas, las funciones de las que dispone el programa y otra información. Luego en primer lugar ejecutamos:

```
$ objdump -d Bomba_NBA_15
$ gdb Bomba_NBA_15
```

Con esto tendremos información sobre el programa así como el programa abierto en el depurador gdb y podremos comenzar a desactivar la bomba. Para ello estableceremos un *breakpoint* en *main* y recorreremos las bomba con **nexti** hasta que la bomba explote. Para tener registro de que escribimos de ahora en adelante estableceremos que sin saber la constraseña escribiremos como contraseña **password=prueba** y el código será **código=0000**.

Tras una primera ejecución la bomba explota en la dirección **0x08048855**, y al fijarnos en la información proporcionada por objdump vemos que en esa dirección se realiza una llamada a <**boom**>. Mirando un poco más arriba vemos que hay dos ordenes que realizan un salto condicional:

```
8048850:    83 f8 3c      cmp $0x3c,%eax
8048853:    7e 05         jle 804885a <main+0x219>
```

Atendiendo a estas dos órdenes vemos que se comparan **0x3c** y **%eax**, y justo depués se hace que si en la comparación %eax es menor que $0x3c, entonces se saltará la llamada a <boom>, en otro caso sigue ejecutando y realiza la llamada que explota la bomba. Para evitar esto estableceremos un breakpoint en la dirección del cmp y cuando al ejecutar alcanzemos dicho breakpoint ejecutaremos **set $eax=0** para que se evite la llamada a <boom>. Con esto podemos eliminar directamente el primer breakpoint.

Ejecutamos de nuevo el programa y vemos que al hacer el *set $eax* se salta la llamada a <boom>seguimos recorriendo el programa utilizando *nexti* hasta encontrar otra llamada que explote la bomba. Al llegar a la direccion **0x080488c6**, la bomba vuelve a explotar y al fijarnos en la información de objdump vemos que se ha producido otra llamada y que nuevamente en las dos lineas anteriores está escrito:

```
80488c1:    83 f8 3c    cmp $0x3c,%eax
80488c4:    7e 05       jle 80488cb <main+0x28a>
```

, que nuevamente compara *%eax* con *$0x3c* y en el caso de que el primero sea menor se salta la llamada a la funcion <boom>. Repetimos el proceso seguido hasta ahora y colocamos un breakpoint que cuando le alcanzemos modificaremos con *set $eax=0*. Nuevamente la bomba explota, esta vez en la dirección **0x080488d2**, sin embargo al mirar un poco más arriba ya no encontramos la misma comparación que hasta ahora, sino que esta vez encontramos:

```
80488cb:    83 7c 24 30 09    cmpl $0x9,0x30(%esp)
80488d0:    74 05             je   80488d7 <main+0x296>
```

, luego se compara *$0x9* y *$0x30(%esp)* y si ambos valores son iguales se saltará la llamada de boom.Estableceremos un breakpoint en el cmpl y modificaremos el valor de *$0x30(%esp)* con **set {int}(0x30+$esp)**, introduciendo así el valor 9 en esta dirección y llegando finalmente al mensaje de bomba desactivada que se llama en la dirección **0x080488d7**.

2. Pasos para averiguar las contraseñas

Para ver cuales son las contraseñas hemos ejecutado la siguiente orden:

```
$ ltrace -i ./Bomba_NBA_2015
```

, que al ejecutar ponemos una contraseña incorrecta y nos muestra las cadenas con las que ha comparado la introducida, dando el siguiente resultado:

```
[0x8048788] gettimeofday(0xffd40a5c, 0)    = 0
[0x8048794] puts("Introduce la contrase\303\261a:
"Introduce la contrasena: = 27
[0x80487b1] fgets(prueba "prueba\n", 100, 0xf7ef05c0) =0xffd40a78
[0x80487c5] strcmp("prueba\n", "Esta es la clave!!\n") = 1
[0x80487ee] strcmp("prueba\n", "Miauuu\n") = 1
[0x804881a] strcmp("prueba\n", "Oh, castitas lilium\n") = 1
[0x8048842] gettimeofday(0xffd40a64, 0) = 0
[0x8048867] printf("Introduce el c\303\263digo: ") = 22
```

, donde podemos observar que este programa tiene tres constraseñas que se consideran correctas. Ahora nos queda por conocer el código por lo que buscaremos la dirección en la que se realiza el scanf y miraremos los cmp que se realizen cerca y como están relacionados. Con lo que nos fijamos en la dirección **0x08048877** se hace la llamada a scanf y podemos observar que justo después encontramos dos cmp en las direcciones **0x08048885** y otro en el **0x08048897** y vemos que en ambos se hace una comparación entre %eax y %edx luego y justo después un jne. Vemos que en ambos cmp el valor de %edx proviene de la misma dirección **0x2c(%esp)**, luego nos fijaremos en los valores de %eax y las instrucciones que implican.Vemos que en el primer cmp %eax=1030 y en el segundo %eax=5700 y que el primero provoca que 0x30(%esp) aumente una unidad y el segundo que se desplaze un bit. El objetivo es conseguir que la dirección **0x30(%esp)** contenga 9. Para ello seguiremos el valor de 0x30(%esp) a lo largo del programa. Iniciamos de nuevo el programa y establecemos un breakpoint en una de las primeras direcciones lo que nos permite ver que el valor de 0x30(%esp) es 4 al inicio del programa. Buscaremos las modificaciones sobre esta dirección para llegar a contener un 9. Donde llegamos a la dirección **0x0804881e**, donde se realiza un shll sobre la dirección, es decir se desplaza un bit, pero esto ocurre si y solo si la contraseña introducida es la tercera. Si seguimos mirando el código vemos que la siguiente modificación sobre la dirección se realiza en función del código introducido y esta pueden ser que incremente en uno o que se desplaze un bit. Claramente nos interese la primera, es decir el código 1030. Luego hemos obtenido que una combinación correcta para la bomba es **password:Oh, castitas lilium código:1030**.

Ahora veamos que es la única posibilidad pues con la primera contraseña al llegar al scanf el valor de 0x30(%esp) es 4 luego si añadimos uno usando el codigo 1030 llegará a 5 y si desplazamos un byte con 5700 será 8 (recordemos que es un código u otro), luego al seguir ejecutando llegaremos al cmpl con $0x9 y la bomba explotará.

Por otro lado con la segunda contraseña pasa exactamente lo mismo que con la primera, con el primer código se obtiene 5 y con el segundo 8 luego la bomba explota al llegar al cmpl.

Otra forma algo más dificir para hallar la constraseña es seguir el rastro de la misma e imprimirla con print (char*) <direccion>.
Luego finalmente hemos llegado a que la única solución para la bomba es **password:Oh, castitas lilium** y **código:1030**