DEPURACIÓN DE MI BOMBA

Para generar el ejecutable, he usado el comando:

gcc -Og JAHC_BOMBA_2021.c -o JAHC_BOMBA_2021 -no-pie -fno-guess-branch-probability

Y para empezar a realizar el análisis de la bomba con el fin de hallar las contraseñas usamos "gdb -tui JAHC_BOMBA_2021" y una vez abierto gdb, usamos layout asm para ver las instrucciones en ensamblador y layout regs para saber las direcciones en las que se guardan los contenidos de los registros. En mi caso, una buena técnica sería empezar buscando todas las líneas de ensamblador en las que se invoque a strncmp, en este caso con layout asm podemos ver que esto ocurre en main+190 y main+236.

main+190: Si el estudiante se fija en el código que sigue a esta orden, en main+195 se comprueba si strncmp ha devuelto 0 o no con la orden test. En el caso de que sí, es porque los string coindían y salta a main+221. Si los string no coinciden, vemos que se ejecutan varias órdenes, entre la cual está en main+216 un printf. Para poder ver los argumentos de printf ponemos un breakpoint: br *main+216. Nos metemos en dicha función con stepi. Como normalmente los dos primeros argumentos se suelen pasar en rdi y rsi, los cuales están en las direcciones 0x1 y 0x402156 respectivamente (según layout regs). Haciendo x/s 0x402156, descubrimos que el argumento pasado a printf era el mensaje "Has caído en mi trampa". Por lo tanto, es inútil analizar lo que ocurre en el strncmp de main+190. Pasamos al otro strncmp, en main+236.

main+236: Si el estudiante se fija en el código que sigue a esta orden, en main+241 se comprueba con test %eax,%eax el resultado de strncmp y si ha dado 0, se salta a main+250 , evitando la llamada a boom en main+245. Esto nos hace pensar que aquí es donde juega su papel la contraseña real. Por lo tanto, ponemos un breakpoint: br *main+236 y hacemos stepi para poder entrar en strncmp. Al igual que antes, vemos los valores de rdi y rsi para ver los valores pasados a strncmp. Para ello, hacemos x/s 0x7fffffffdf00 (rdi) y x/s 0x7fffffffdee0 (rsi), y obtenemos el valor del string que hemos introducido en ejecución y "uncincosonseiscreditos" respectivamente. Por lo tanto, "uncincosonseiscreditos" debe ser la contraseña. Usamos run para ejecutar de nuevo el programa, introducimos dicha contraseña y vemos que funciona. Ya solo nos queda ver el pin.

Para hallar el pin, vamos a buscar en el código ensamblador las siguientes llamadas a boom. Vemos que están en main+400 y main+436. El primer boom se ejecuta justo después de una instrucción cmpl (main+390) y je (main+398). También desde main+366 se salta a main+390 en caso de que el cmpl de main+358 de un resultado concreto. Como ambas llamadas cmp son las que conducen a la llamada a boom, estudiamos qué comparan. En ambas se compara el registro edx con un número, 0x778 y 0x57e respectivamente. Como están en hexadecimal, vemos los valores en decimal haciendo "p /d 0x778" que es 1912 y "p /d 0x57e" que es 1406. Volvemos a realizar la ejecución con gdb usando run y probamos ambos números, viendo que el último, 1406, logra que se desactive la bomba.