CSE 5474 — Software Security Spring 2022

CTF5 Writeups

format string CTF *Tron Tian*

1 Overview

In CTF 5, we use format string to leak some useful information such as buffer address to exploit the programs using some useful tools and techniques such as ROP, buffer overflow.

2 Steps of how to Solve the Challenges

2.1 Level 1

In level 1, we built our payload by using FmtStr method provided by pwntool library and exec_fmt method provided by ctf5 instruction. Then we used printf() function to get got(Global Offset table) and calculate address of win function to make sure our payload is inserted to correct position.

2.2 Level 2

In level2, we first need to found got table from exit() function. Then we got address of win function by using elf.symbols() method provided by pwntools. Then we intiialize autofmt class as we did in level1. Then we wrote address win() function to got table. Finally, we made our payload and send it to the vulnerable process.

2.3 Level 3

In level3, we first use the buffer address provided by the program itself to get address instruction pointer. Then we built shellcode using setuid(0) + /bin/sh. We need to place our shellcode in buffer and using format string vulnerability to overflow return address to execute our shellcode.

2.4 Level 4

Level4 is similar to level3, but we need to bypass the canary protection so that we don't change value of the canary which may cause the program to fail. Before sending our actual payload, we need to find canary's location and bypass it.

2.5 Level 5

We can use the skeleton of level 4 to solve level5. However, the address information was found manually. Brute forcing and finding the address and stack information sometimes better than nothing.

2.6 Level 6

PIE was enabled in level6, so we cannot directly find the correct address and return to it. However, the last two digits of the address remain same(0x7d in this case). Brute forcing was used in this challenge to get correct address.

2.7 Level 7

Level7 is similar to level6. The digits remaining same this time is 0x0aa7. Then we added 138 bytes(130 bytes from rbp to rsp, and 8bytes of rbp) to the payload. It needs to be noticed that we need to bypass the if statement to exploit the program.

2.8 Level 8

Level8 is similar to level7, however we need to pass 1 and 3 as parameters to win() function.

2.9 Level 9

Level9 is the most difficult one to solve. First we built the first rop to leak libc address. Since we cannot do two rops in the same time, we migrate stack position to build the secod rop. The position of migrated stack position is a section of memory that allows writing between func and main function. Since read()function is a function needing three parameters, we use ret2csu to build rop of three paramenters fucntion.Then we use setuid and /bin/sh to build our payload then exploit the program.

2.10 Level 10

Since NX was not enabled, we can easily inject our shellcode and execute it by using information of given return address.

2.11 Level 11

PIE is enable in level11, so the address varies every time. The key to solve this challenge is finding our the correct base address. The challenge was solved by brute forcing to find the return address, then the offset(0x0db0) was found using ghidra.