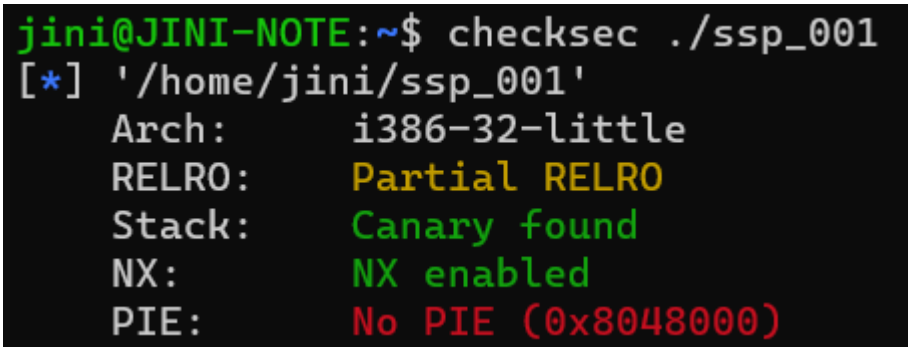
ssp_001

ssp_001.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");
}
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;
        }
    }
}
```

보호기법 확인



- 32bit 바이너리이다.
- Canary 가 적용되어 있다.
- NX 방어기법이 적용되어 있다.

코드 분석

get_shell() 함수

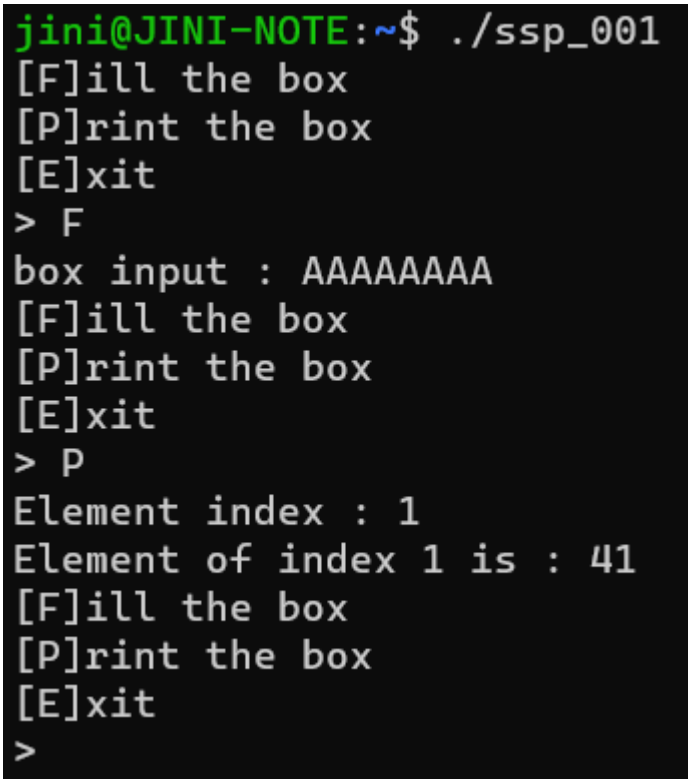
```
void get_shell() {
    system("/bin/sh");
}
```

get_shell() 함수를 호출하면 /bin/sh 쉘을 획득할 수 있다.

print_box() 함수

```
void print_box(unsigned char *box, int idx) {
    printf("Element of index %d is : %02x\n", idx, box[idx]);
}
```

box 의 idx 에 해당하는 값을 hex 형태로 출력해준다.



임의의 값 A 를 입력하여 print_box 가 호출되었을 때 A 에 해당하는 hex(0x41) 이 출력 되는 것을 알 수 있다.

이를 통해 정해진 buf 의 크기보다 더 많은 값을 idx 를 통해 입력하여 데이터를 읽을 수 있다.

따라서 buf 뒤의 4byte 인 canary의 값을 얻을 수 있다.

void menu() 함수

```
void menu() {
    puts("[F]ill the box");
    puts("[P]rint the box");
    puts("[E]xit");
    printf("> ");
}
```

단순히 출력만 하는 함수라고 볼 수 있다.

int main() 함수

```
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] ) {
```

initialize() 함수는 가볍게 생각하면 일정 시간이 지나면 프로그램을 종료시키는 함수이다.

- box = 0x40
- name = 0x40
- select = 2
- idx = 0
- name_len = 0

이렇게 각자 할당되어 있다.

while(1) 함수로 반복문을 실행한다.

- menu() 함수를 불러 3가지의 선택지를 출력한다.
- read() 함수로 입력 값을 읽어 switch 문을 동작 시킨다.

read() 함수로 읽었을 때 select 의 크기는 2로 선언되어 있고, 2까지 읽을 수 있다.

select[0] 으로 첫 번째 글자를 읽어 switch 문을 동작 시킨다.

→ 즉, menu() 함수를 통해 F, P, E 중에서 하나를 입력되게 하고 입력 값을 2byte 까지 입력할 수 있어 FF, PP, EE 등 여러 조합으로 입력할 수 있지만, select[0] 에 의해 한 글자만 인식된다.

case ‘F’

```
case 'F':
     printf("box input : ");
     read(0, box, sizeof(box));
     break;
```

- F 를 입력했을 때 box in put: 이라는 문자열을 출력한다.

- box 를 입력 받고 끝난다.
- box 의 크기는 64byte 이고 (int(0x40)), sizeof(box) 로 64byte 까지 동일하게 제한된다.
    - → 따라서 취약점이 발생하지 않는다고 볼 수 있다.

case ‘P’

```
case 'P':
     printf("Element index : ");
     scanf("%d", &idx);
     print_box(box, idx);
     break;
```

- P 를 입력했을 때 Element index: 이라는 문자열을 출력한다.
- idx 를 입력 받고 print_box() 함수에 box 와 idx 를 인자로 넘긴다.
- idx 는 int idx = 0 으로 설정되어 있다.
- int 형으로 들어갈 수 있게 숫자를 입력하면 print_box() 함수가 실행된다.

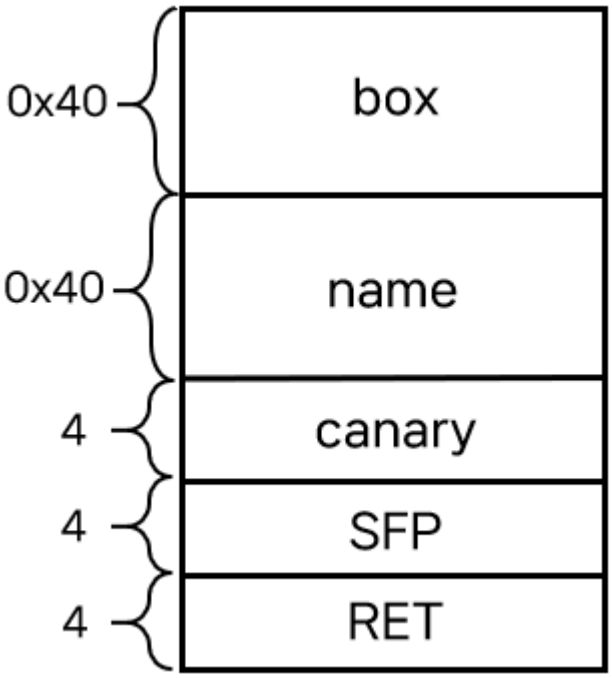
case ‘E’

```
case 'E':
     printf("Name Size : ");
     scanf("%d", &name_len);
     printf("Name : ");
     read(0, name, name_len);
     return 0;
  default:
     break;
```

- E 를 입력했을 때 Name Size: 를 출력한다.
- name_len 을 입력 받는다.
- int 형으로 설정되었기 때문에 숫자를 입력해야 한다.
- 입력 후 Name: 을 출력한다.
- 64byte 로 설정된 name 을 입력 받고 앞에서 입력한 name_len까지 읽는다.

즉, buf 공간은 64byte 로 선언했지만 공격자가 원하는 크기까지 읽을 수 있기 때문에 Buffer over Flow 가 발생할 수 있다.

간단한 구조를 살펴보겠다.



1. case P 에서 idx 를 입력 받아 box 에서 원하는 idx 까지 출력 시킬 수 있다.
2. box 부터 canary 까지인 0x80, 즉, 128 부터 시작해서 129, 130, 131 까지 읽는다.
3. 4byte 의 canary 값을 얻을 수 있다.

익스플로잇이 일어나는 구간은 case E 이다.

문제 풀이

name_len 에 원하는 크기만큼 읽을 수 있도록 값을 정할 수 있다.

name 의 buf 구간부터 입력할 수 있다.

> ✔️ name 에 64byte (name 크기) + 4byte (P에서 얻어낸 canary) + 4byte (SFP) + 4byte (get_shell 의 주소)

공격자가 Buffer over Flow 를 발생 시켜 RET의 주소까지 올라가 원하는 값으로 조작하여 get_shell 함수를 실행 시킬 수 있다.

GDB 정적 분석 - get_shell 함수

```
pwndbg> print get_shell
$1 = {<text variable, no debug info>} 0x80486b9 <get_shell>
```

0x80486b9 가 get_shell 주소임을 확인했다.

카나리 릭

……gdb 사용에 오류가 떠서 해결하는 중입니다……..

해결하고 다시 올리도록 하겠습니다…

```python
from pwn import *

p = remote("")

get_shell = 0x80486b9
canary = ''

p.interactive()
```

지금까지의 익스플로잇 코드이다.