[What is system hacking]

버퍼 오버플로우

2022. 05. 04

SCP 2학년 이예준

Index

- ✓ 버퍼 오버플로우란?
- ✓ 스택(stack)
- ✓ 함수 호출과 스택 프레임
- ✓ 버퍼 오버플로우 실습
- ✓ 코드 실행 결과
- ✓ 코드취약점 분석
- ✓ 프로그램 흐름 변경
- ✓ 이를 방지하기 위해서

버퍼 오버플로우란?

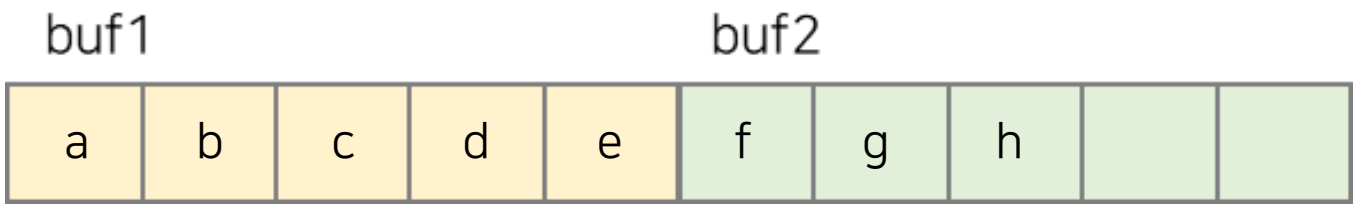
✓ 버퍼

임시 저장 공간을 의미, A와 B가 입출력을 수행하는데 있어서 속도 차이를 극복하기 위해 사용

✓ 버퍼 오버플로우

이 프로그램이 실행될 때 이 메모리 공간에 버퍼의 크기보다 더 많은 입력을 받는다면 버퍼 오버플로우가 발생

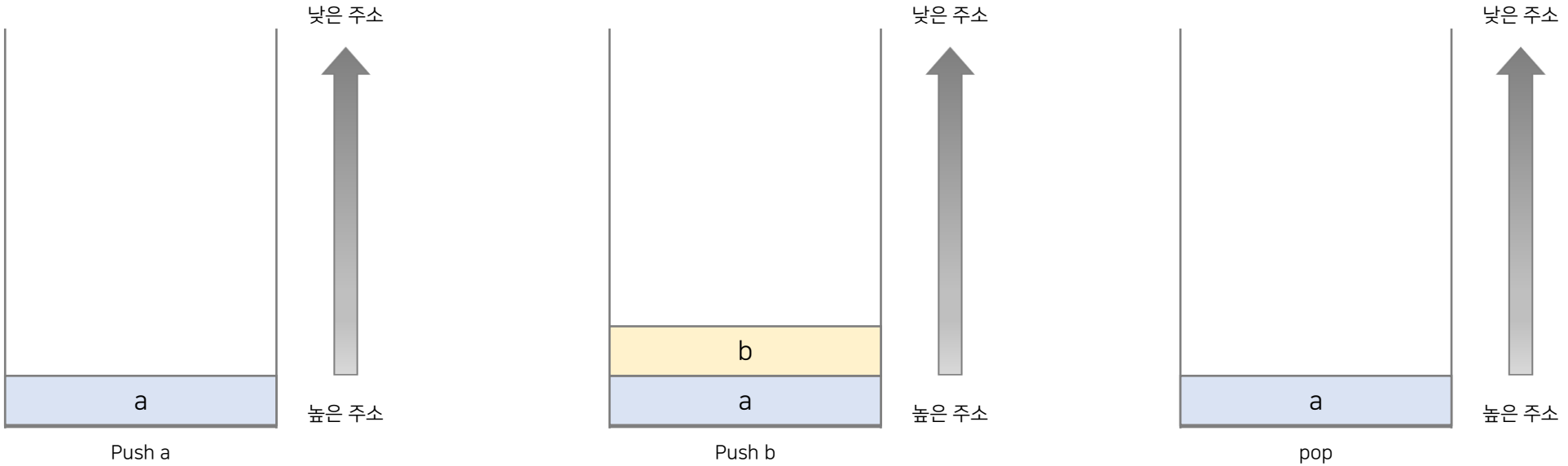
이 경우 해커가 특정 메모리 값을 임의로 변조할 수 있기 때문에 취약점이 된다.



[맥주 조아해?]

스택(stack)

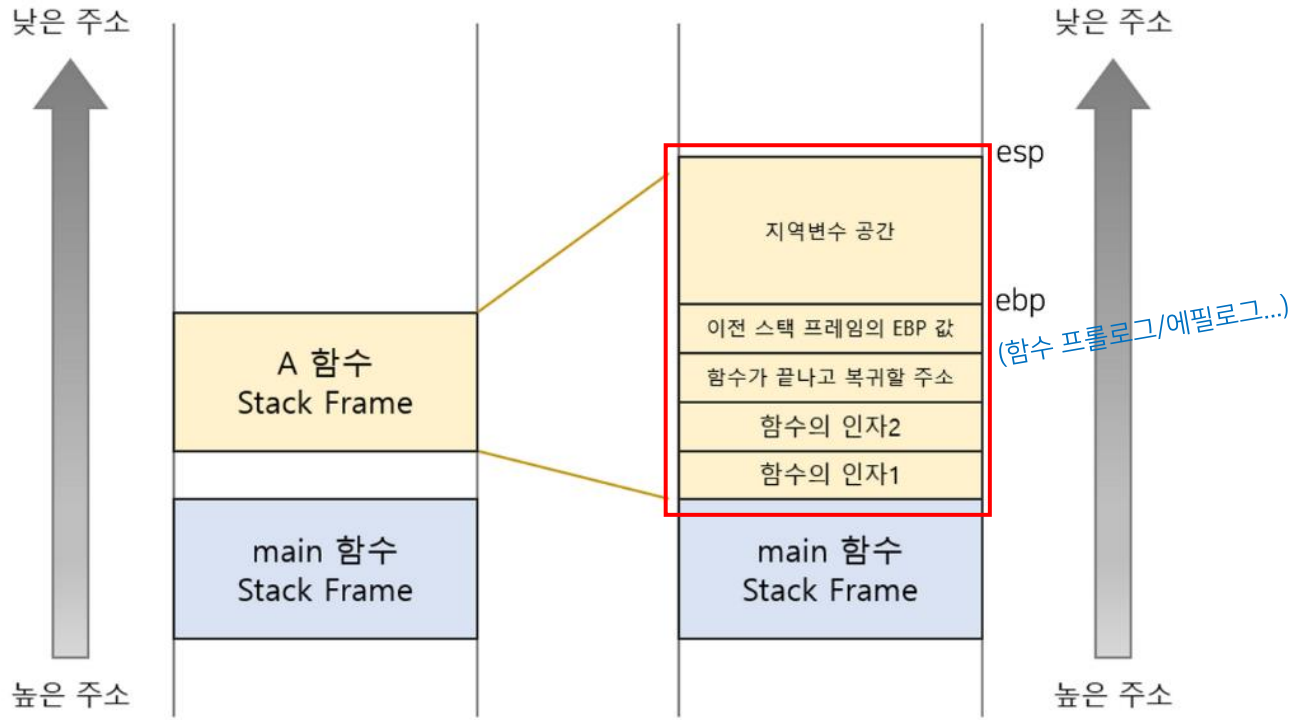
- ✓ 메모리의 스택 영역은 **함수의 호출과 관계되는 지역변수, 매개변수, 리턴 값등의 임시데이터를 저장합니다.**
- ✓ 스택이란 단어는 '차곡 차곡 쌓여진 더미'를 의미합니다.
- ✓ LIFO(Last In First Out, 후입선출) 구조라고도 합니다.



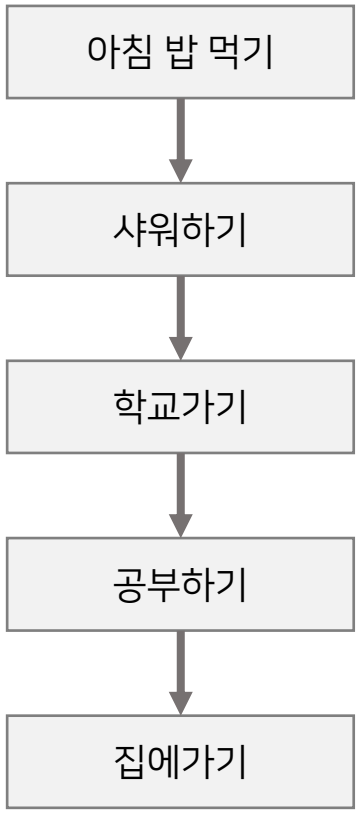
[프링글스 조아해?]

함수 호출과 스택 프레임

✓ Main 함수 시작 -> a 함수 시작 -> a 함수 종료 -> main 함수 종료

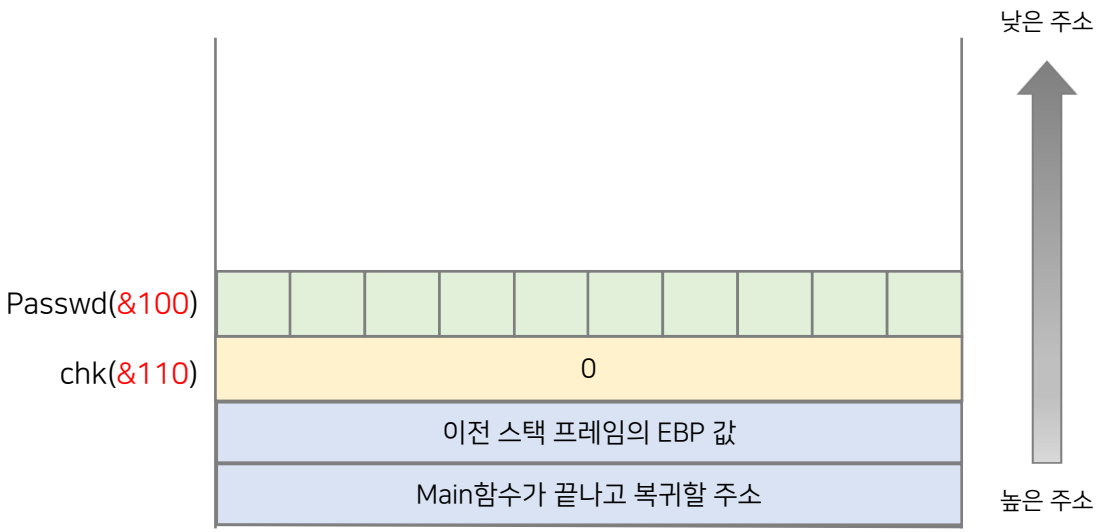


✓ 인간의 흐름



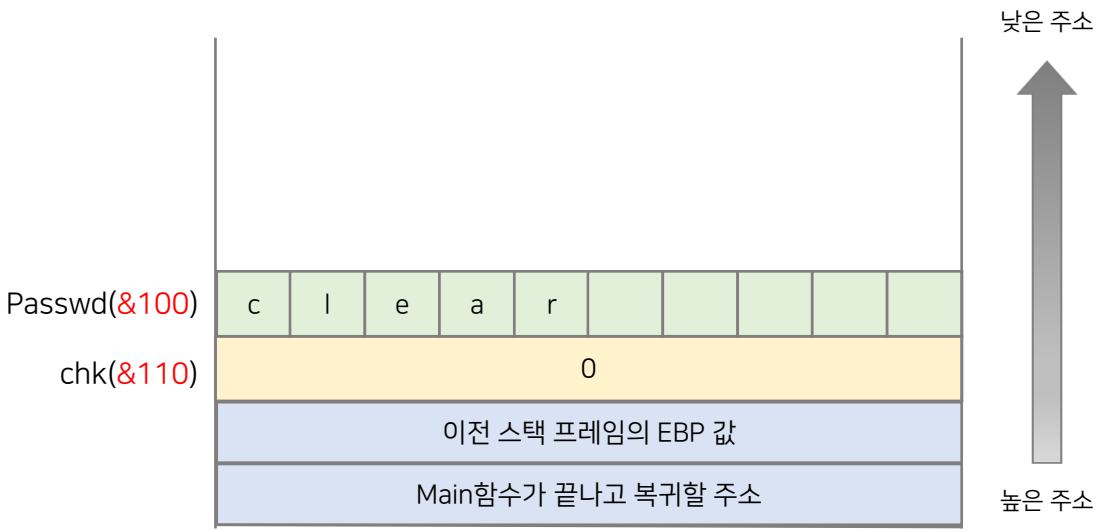
버퍼 오버플로우 실습

```c
//name : bof.c
//compile : gcc -o bof bof.c -fno-stack-protector

#include<stdio.h>
int main(){
    char chk = '0';
    char passwd[10];

    scanf("%s",passwd);

    if (*passwd == *"clear") chk = '1';

    if (chk == '1'){
        printf("success!");
    } else{
        printf("fail!");
    }

    return 0;
}
```



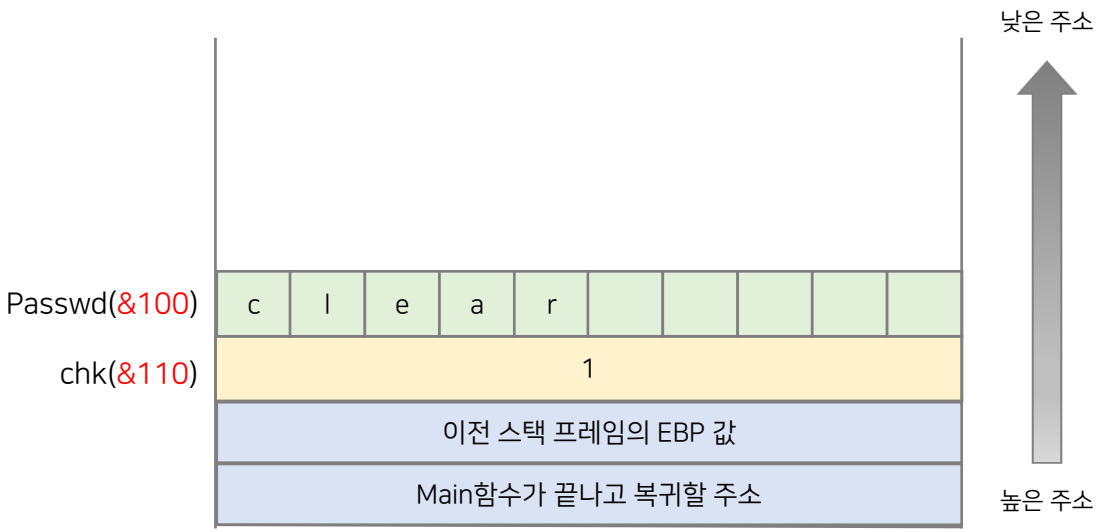
버퍼 오버플로우 실습

```c
//name : bof.c
//compile : gcc -o bof bof.c -fno-stack-protector

#include<stdio.h>
int main(){
    char chk = '0';
    char passwd[10];

    scanf("%s",passwd);

    if (*passwd == *"clear") chk = '1';

    if (chk == '1'){
        printf("success!");
    } else{
        printf("fail!");
    }

    return 0;
}
```



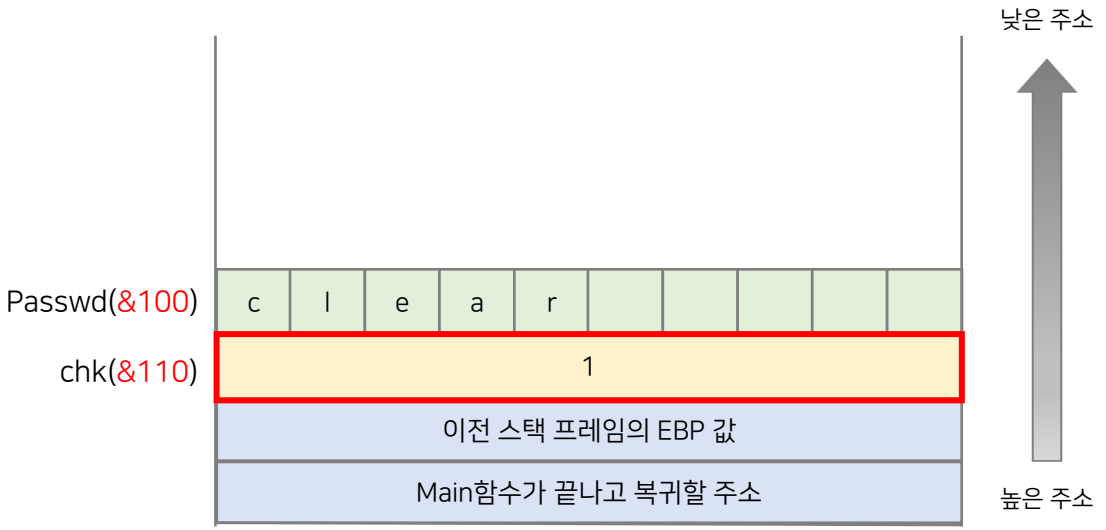
버퍼 오버플로우 실습

```c
//name : bof.c
//compile : gcc -o bof bof.c -fno-stack-protector

#include<stdio.h>
int main(){
    char chk = '0';
    char passwd[10];

    scanf("%s",passwd);

    if (*passwd == *"clear") chk = '1';

    if (chk == '1'){
        printf("success!");
    } else{
        printf("fail!");
    }

    return 0;
}
```



버퍼 오버플로우 실습

```c
//name : bof.c
//compile : gcc -o bof bof.c -fno-stack-protector

#include<stdio.h>
int main(){
    char chk = '0';
    char passwd[10];

    scanf("%s",passwd);

    if (*passwd == *"clear") chk = '1';

    if (chk == '1'){
        printf("success!");
    } else{
        printf("fail!");
    }

    return 0;
}
```



코드 실행 결과

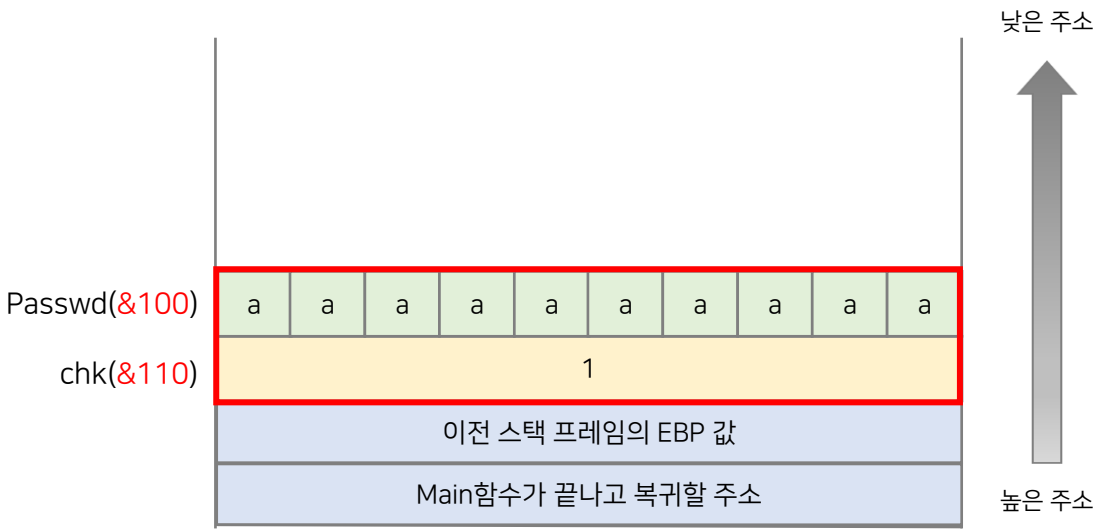
```c
//name : bof.c
//compile : gcc -o bof bof.c -fno-stack-protector

#include<stdio.h>
int main(){
    char chk = '0';
    char passwd[10];

    scanf("%s",passwd);

    if (*passwd == *"clear") chk = '1';

    if (chk == '1'){
        printf("success!");
    } else{
        printf("fail!");
    }

    return 0;
}
```

< 실행 결과 >

```
 yejun  ~/study/stack  ./bof
clear
success!%
 yejun  ~/study/stack  ./bof
hello
fail!%
```

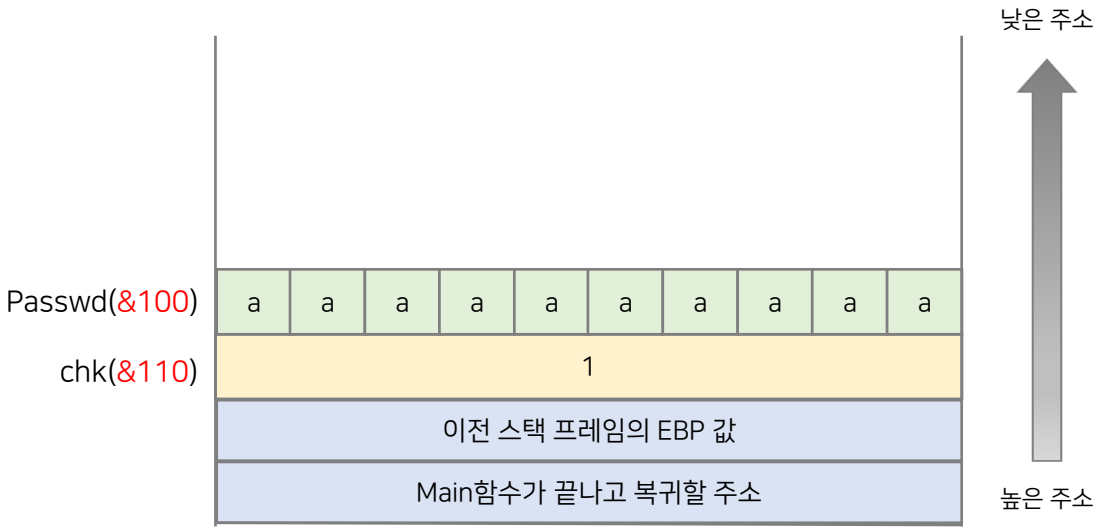
코드 취약점 분석

```c
//name : bof.c
//compile : gcc -o bof bof.c -fno-stack-protector

#include<stdio.h>
int main(){
    char chk = '0';
    char passwd[10];

    scanf("%s",passwd);  // aaaaaaaaaa1 입력

    if (*passwd == *"clear") chk = '1';

    if (chk == '1'){
        printf("success!");
    } else{
        printf("fail!");
    }

    return 0;
}
```



코드 취약점 분석

```c
//name : bof.c
//compile : gcc -o bof bof.c -fno-stack-protector

#include<stdio.h>
int main(){
    char chk = '0';
    char passwd[10];

    scanf("%s",passwd);    // aaaaaaaaaa1 입력

    if (*passwd == *"clear") chk = '1';    // false

    if (chk == '1'){
        printf("success!");
    } else{
        printf("fail!");
    }

    return 0;
}
```



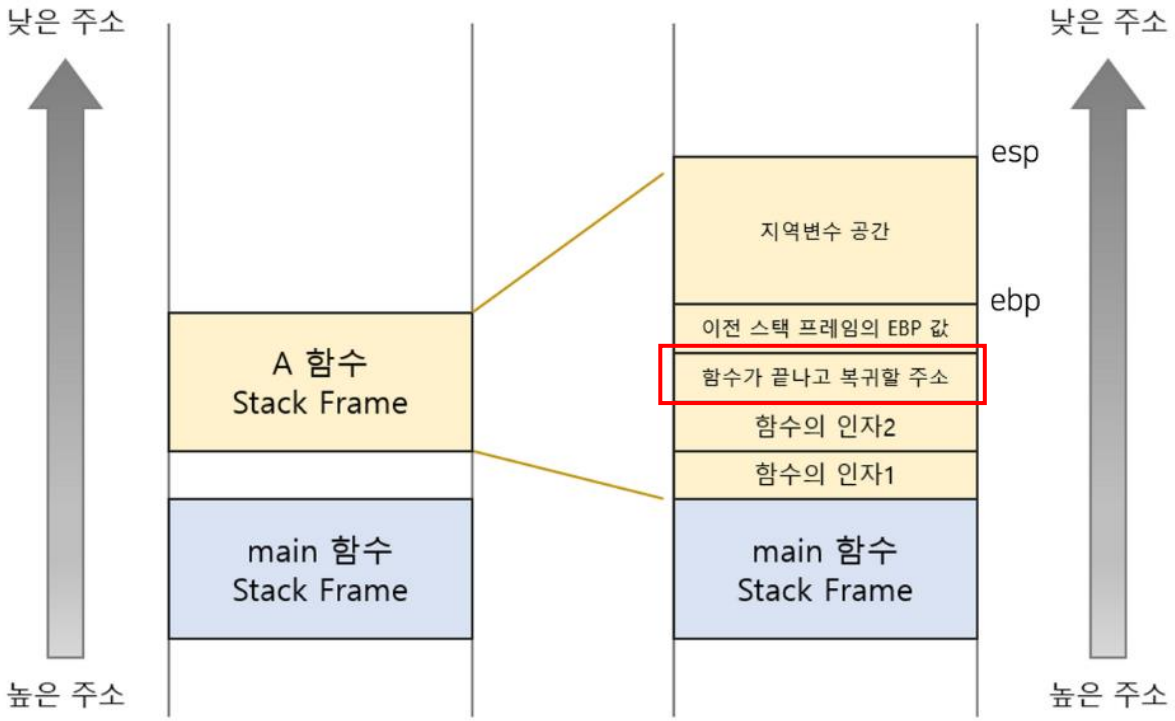
코드 취약점 분석

```c
//name : bof.c
//compile : gcc -o bof bof.c -fno-stack-protector

#include<stdio.h>
int main(){
    char chk = '0';
    char passwd[10];

    scanf("%s",passwd);    // aaaaaaaaaa1 입력

    if (*passwd == *"clear") chk = '1';    // false

    if (chk == '1'){
        printf("success!");
    } else{
        printf("fail!");
    }

    return 0;
}
```

< 실행 결과 >

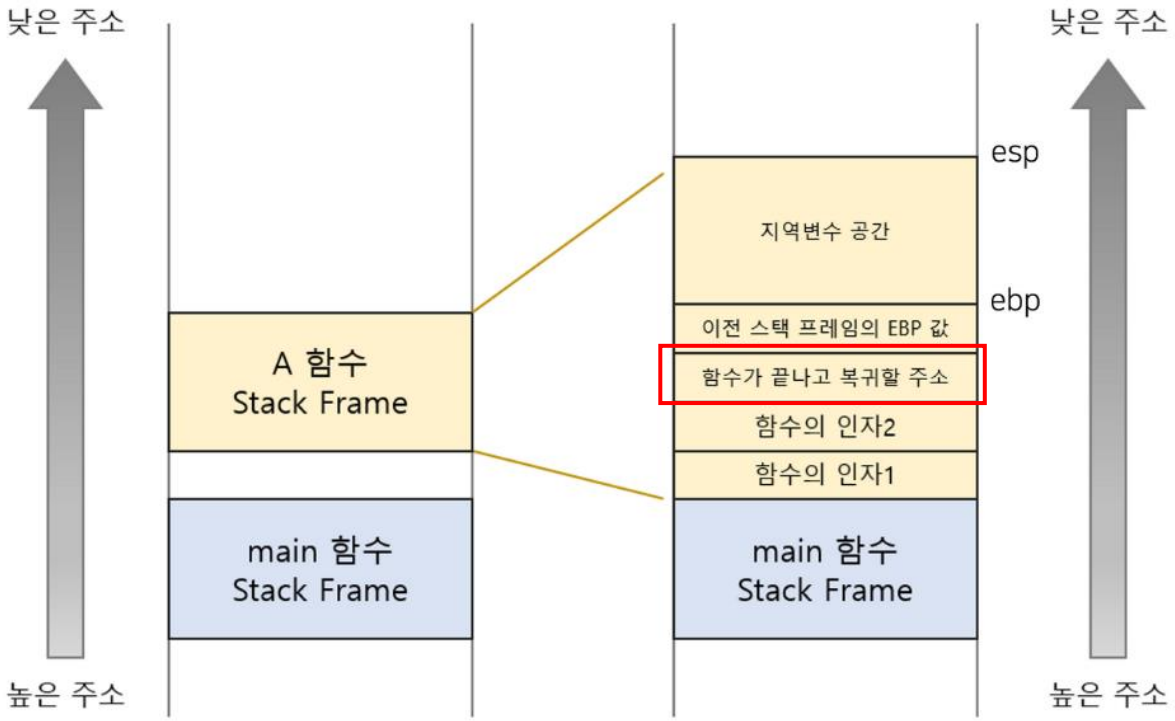
프로그램 흐름 변경?

✓ Main 함수 시작 -> a 함수 시작 -> a 함수 종료 -> ???

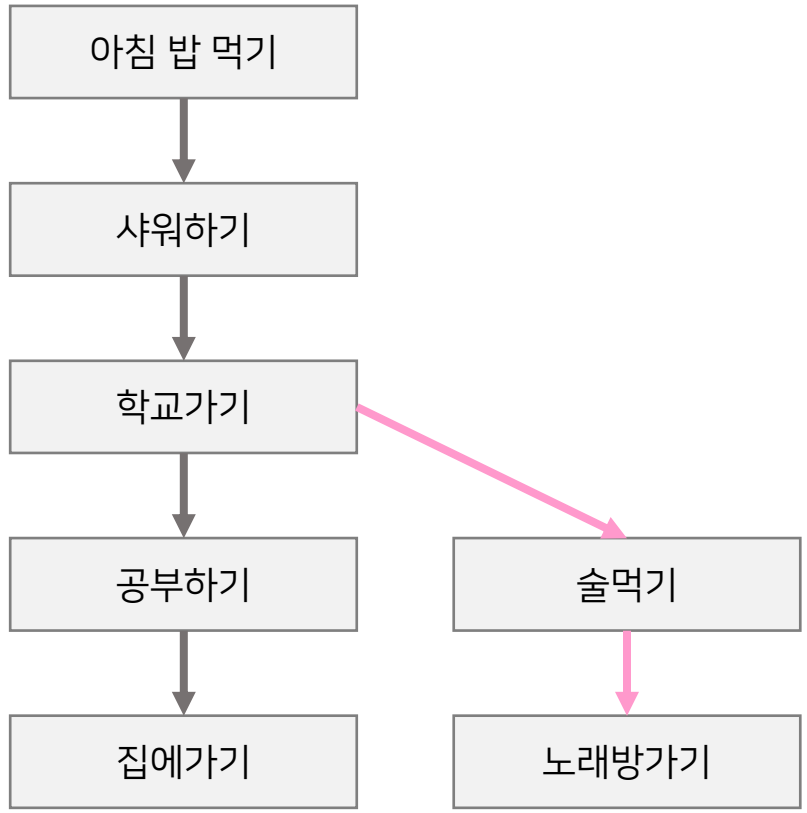


프로그램 흐름 변경

✓ Main 함수 시작 -> a 함수 시작 -> a 함수 종료 -> ???



✓ 인간의 흐름 변경.. (인간의 취약점)

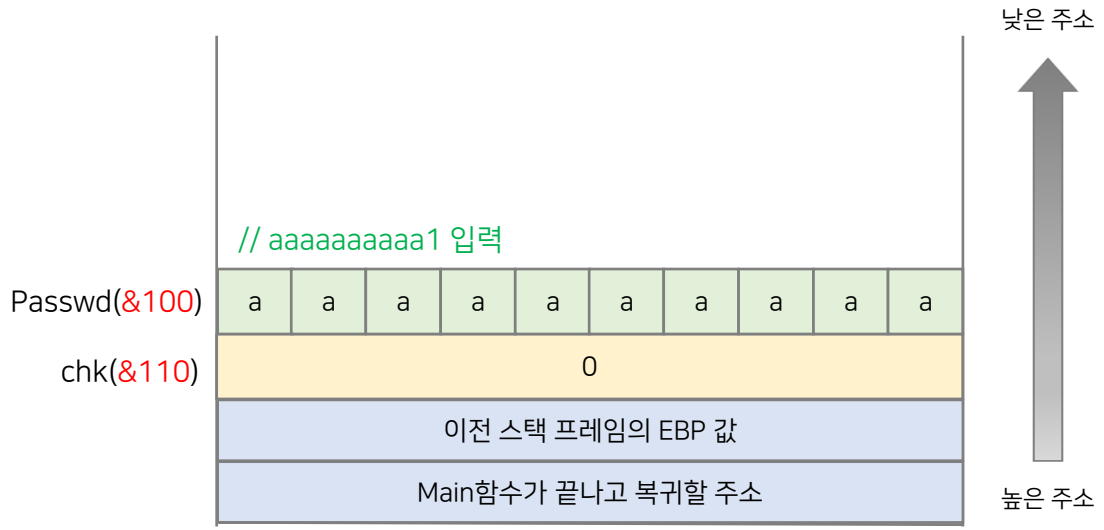


이를 방지하기 위해서..

프로그램 입력 길이 제한

- ✓ read(0,&buf,strlen(buf))
- ✓ scanf_s("%s",&buf,strlen(buf));
- ! gets(&buf)

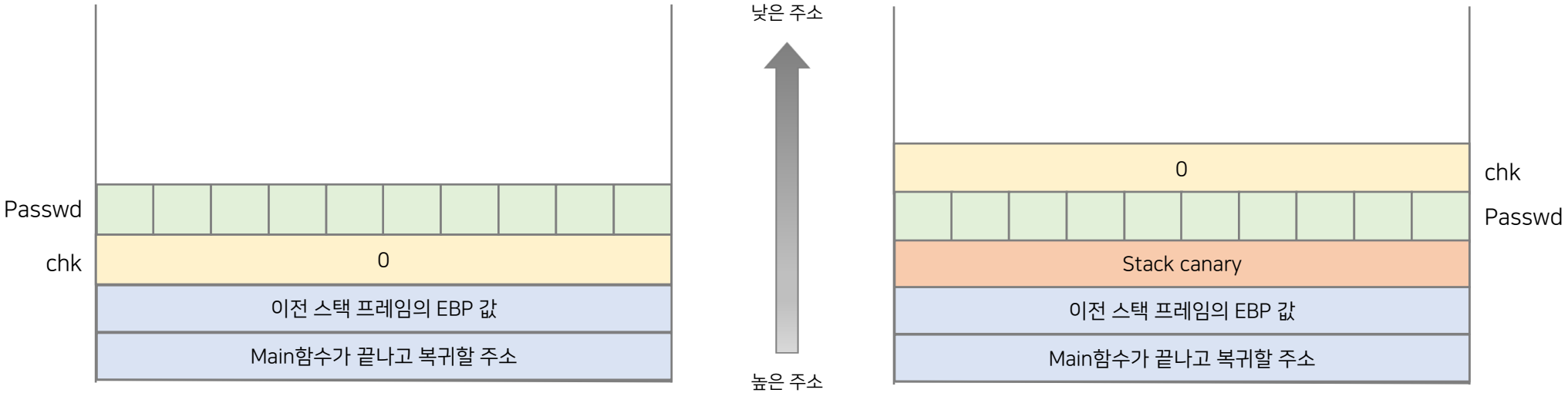
입력을 n개만 할 수 있도록



이를 방지하기 위해서..

메모리 보호기법 : SSP (Stack Smashing Protector)

- ✓ 로컬 변수 재배치
- ✓ 로컬 변수 전에 포인터 배치
- ✓ canary 삽입



[SSP 적용 전]

[SSP 적용 후]

EZ CTF | Beginner Friendly

금요일, 06 5 2022, 16:00 UTC — 토요일, 07 5 2022, 16:00 UTC

On-line

A EZ CTF event.

Format: Jeopardy

Official URL: https://ez.ctf.cafe/

Future weight: 0.00

Rating weight: 0.00

Event organizers

- MY_LITTLE_PWNIES

We are hosting a CTF to test our new CTF hosting service. This is our first test so we are aiming at beginners, our aim is to get more and more people into the CTFs spirit as we love to compete and learn new things! We will have categories such as web, crypto, osint and many more!