Segfault

- LOB에서는 입력된 값이 버퍼오버플로우를 일으키게 되면

> segmentation fault (core dumped)

라는 메시지를 띄우며 코어파일을 생성한다.

- 그러나 항상 띄우는 것이 아니다.

```
/*
        The Lord of the BOF : The Fellowship of the BOF
        - cobolt
        - small buffer
*/

int main(int argc, char *argv[])
{
    char buffer[16];
    if(argc < 2){
        printf("argv error\n");
        exit(0);
    }
    strcpy(buffer, argv[1]);
    printf("%s\n", buffer);
}
```

- 이러한 코드가 있다. 그러면 스택에는 | 높은주소 | |-| |buffer[16]| |SFP| |RET|

이런식으로 자리를 잡게 된다. 여기서 buffer의 크기보다 입력된 값의 크기가 더 크면 BOF가 일어나게 된다. 입력한 값이 16byte의 버퍼를 채우면 정상적으로 종료된다. 입력한 값이 16byte를 넘어가게 되면 어떤일이 일어나는지 gdb를 사용하여 알아보았다.

```
(gdb) r `python -c 'print "A"*17'`
Starting program: /home/gremlin/tmp/cobalt `python -c 'print "A"*17'`

Breakpoint 1, 0x8048468 in main ()
(gdb) x/50x $esp
0xbffffb08:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffb18:     0xbfff0041      0x400309cb      0x00000002      0xbffffb64
0xbffffb28:     0xbffffb70      0x40013868      0x00000002      0x08048380
0xbffffb38:     0x00000000      0x080483a1      0x08048430      0x00000002
0xbffffb48:     0xbffffb64      0x080482e0      0x080484ac      0x4000ae60
0xbffffb58:     0xbffffb5c      0x40013e90      0x00000002      0xbffffc5e
0xbffffb68:     0xbffffc77      0x00000000      0xbffffc89      0xbffffcab
0xbffffb78:     0xbffffcb5      0xbffffcc3      0xbffffce2      0xbffffcf2
0xbffffb88:     0xbffffd0a      0xbffffd27      0xbffffd32      0xbffffd40
0xbffffb98:     0xbffffd83      0xbffffd96      0xbffffdab      0xbffffdbb
0xbffffba8:     0xbffffdc8      0xbffffde7      0xbffffe00      0xbffffe0b
0xbffffbb8:     0xbffffe18      0xbffffe20      0x00000000      0x00000003
0xbffffbc8:     0x08048034      0x00000004
```

- 0xbfff0041 이 부분에 41뒤에 00이 붙는 것을 보아 문자열의 끝을 알리는 \x00값이 추가된 것을 추정했다.

```
(gdb) r `python -c 'print "A"*18'`
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/gremlin/tmp/cobalt `python -c 'print "A"*18'`

Breakpoint 1, 0x8048468 in main ()
(gdb) x/50x $esp
0xbffffb08:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffb18:     0xbf004141      0x400309cb      0x00000002      0xbffffb64
0xbffffb28:     0xbffffb70      0x40013868      0x00000002      0x08048380
0xbffffb38:     0x00000000      0x080483a1      0x08048430      0x00000002
0xbffffb48:     0xbffffb64      0x080482e0      0x080484ac      0x4000ae60
```

```
0xbffffb58:     0xbffffb5c      0x40013e90      0x00000002      0xbffffc5d
0xbffffb68:     0xbffffc76      0x00000000      0xbffffc89      0xbffffcab
0xbffffb78:     0xbffffcb5      0xbffffcc3      0xbffffce2      0xbffffcf2
0xbffffb88:     0xbffffd0a      0xbffffd27      0xbffffd32      0xbffffd40
0xbffffb98:     0xbffffd83      0xbffffd96      0xbffffdab      0xbffffdbb
0xbffffba8:     0xbffffdc8      0xbffffde7      0xbffffe00      0xbffffe0b
0xbffffbb8:     0xbffffe18      0xbffffe20      0x00000000      0x00000003
0xbffffbc8:     0x08048034      0x00000004


(gdb) r `python -c 'print "A"*19'`
Starting program: /home/gremlin/tmp/cobalt `python -c 'print "A"*19'`

Breakpoint 1, 0x8048468 in main ()
(gdb) x/50x $esp
0xbffffb08:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffb18:     0x00414141      0x400309cb      0x00000002      0xbffffb64
0xbffffb28:     0xbffffb70      0x40013868      0x00000002      0x08048380
0xbffffb38:     0x00000000      0x080483a1      0x08048430      0x00000002
0xbffffb48:     0xbffffb64      0x080482e0      0x080484ac      0x4000ae60
0xbffffb58:     0xbffffb5c      0x40013e90      0x00000002      0xbffffc5c
0xbffffb68:     0xbffffc75      0x00000000      0xbffffc89      0xbffffcab
0xbffffb78:     0xbffffcb5      0xbffffcc3      0xbffffce2      0xbffffcf2
0xbffffb88:     0xbffffd0a      0xbffffd27      0xbffffd32      0xbffffd40
0xbffffb98:     0xbffffd83      0xbffffd96      0xbffffdab      0xbffffdbb
0xbffffba8:     0xbffffdc8      0xbffffde7      0xbffffe00      0xbffffe0b
0xbffffbb8:     0xbffffe18      0xbffffe20      0x00000000      0x00000003
0xbffffbc8:     0x08048034      0x00000004
```

- 사실상 0x00값을 포함하여 SFP를 덮은거로 판단된다.

```
(gdb) r `python -c 'print "\x41"*17'`
Starting program: /home/gremlin/tmp/cobalt `python -c 'print "\x41"*17'`

Breakpoint 1, 0x8048468 in main ()
(gdb) x/50x $esp
0xbffffb08:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffb18:     0xbfff0041      0x400309cb      0x00000002      0xbffffb64
0xbffffb28:     0xbffffb70      0x40013868      0x00000002      0x08048380
0xbffffb38:     0x00000000      0x080483a1      0x08048430      0x00000002
0xbffffb48:     0xbffffb64      0x080482e0      0x080484ac      0x4000ae60
0xbffffb58:     0xbffffb5c      0x40013e90      0x00000002      0xbffffc5e
0xbffffb68:     0xbffffc77      0x00000000      0xbffffc89      0xbffffcab
0xbffffb78:     0xbffffcb5      0xbffffcc3      0xbffffce2      0xbffffcf2
0xbffffb88:     0xbffffd0a      0xbffffd27      0xbffffd32      0xbffffd40
0xbffffb98:     0xbffffd83      0xbffffd96      0xbffffdab      0xbffffdbb
0xbffffba8:     0xbffffdc8      0xbffffde7      0xbffffe00      0xbffffe0b
0xbffffbb8:     0xbffffe18      0xbffffe20      0x00000000      0x00000003
0xbffffbc8:     0x08048034      0x00000004
(gdb) q
```

- HEX값으로 전달해도 0x00이 들어간다.

Why?

- 32bit운영체제를 기준으로 buffer가 16byte일 때 20byte의 문자열을 입력하게 되면 0x00까지 포함하여 사실상 21byte를 채우는 것으로 보인다.
- 그러면 buffer 16byte를 채우고 sfp 4byte, RET 1byte를 침범한다.
  문자열의 끝값을 나타내는 0x00값이 RET주소를 침범하게 되어 segmentation fault를 띄우고 core를 생성하는 듯 하다.