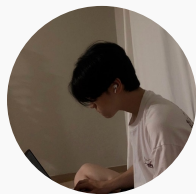


Pwnable? 이게 무슨 말이죠?

시스템 해킹, 주제넘게 아는 척 해봅니다!



송지호

GDSC Soongsil Server Part.

jjhojiho2003@gmail.com

```
filterByOrg = filterByOrg ? study.lead_organization === filterByOrg : true  
filterByStatus = filterByStatus ? study.status === filterByStatus : true  
if (!filterByOrg || !filterByStatus || !matchStatus) {
```

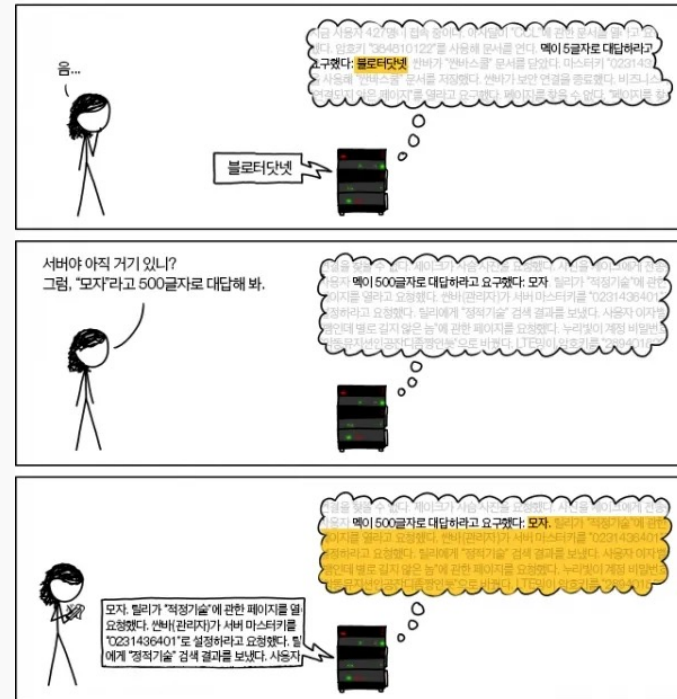
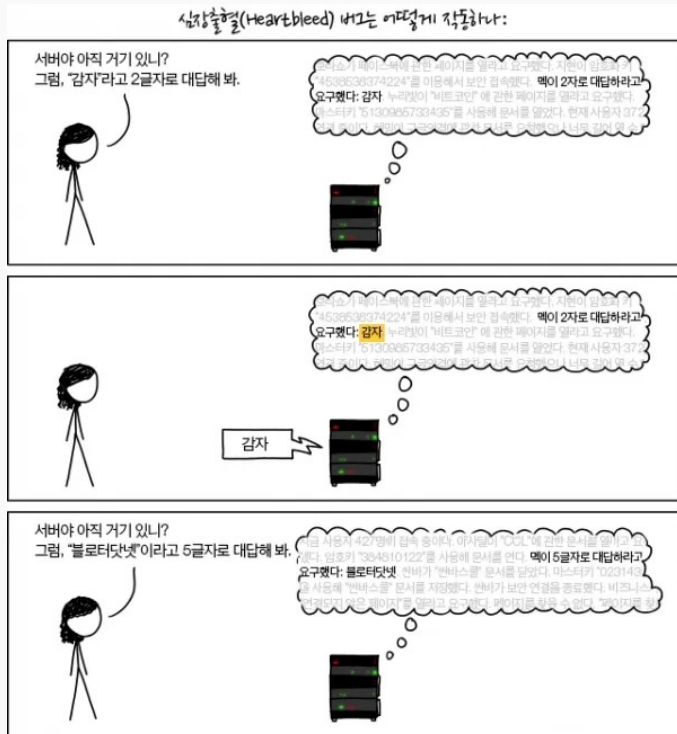
```
function filterStudies({ studies, filterByOrg = false, filterByStatus = false, filterByLeadOrganization = false }) {  
  return studies.filter(study => {  
    return filterByOrg || filterByStatus || filterByLeadOrganization
```

Pwnable이란?

- leet → 영미권 사이트에서 사용하는 인터넷 은어의 총칭
- pwn → owned를 급하게 쓰다가 o옆에 있는 p를 입력하여 오타가 난 것.
의역하자면 ‘털었어’, ‘관광보냈어’

컴퓨터의 취약점을 찾아 악용할 수 있는가?, Pwnable?

시큐어 코딩의 중요성 출처 xkcd



Pwnable, 시스템 해킹

- 운영 체제나 소프트웨어, 하드웨어에 내제 된 보안 취약점을 해킹 하는 것



System Hacking



Reverse Engineering



Web Hacking



Cryptography



Mobile Hacking



일반적인 해킹 방어 대회 분야 분류 (출처 [드림핵](#))

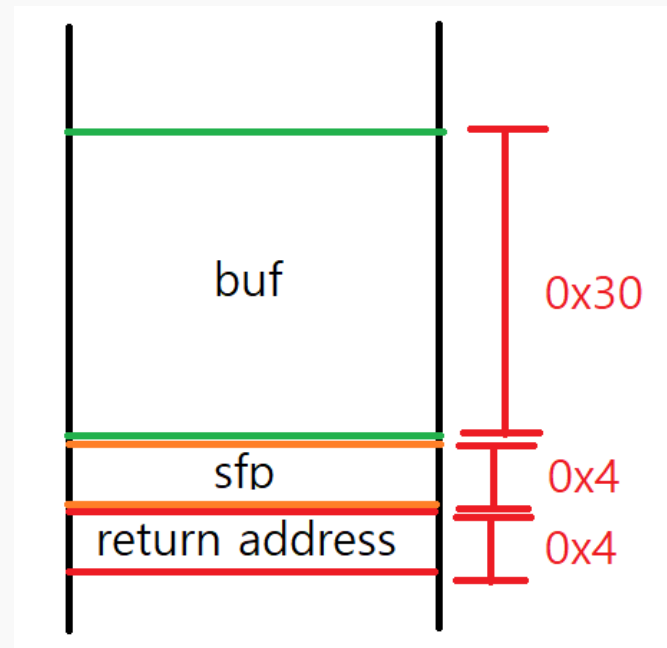
Pwnable을 공부하려면?

- 운영체제나 소프트웨어, 하드웨어에 내제 된 보안 취약점
→ 운영체제나 소프트웨어, 하드웨어를 공부하면 됨!
- 운영체제 이론 → Windows API, 리눅스 커널 → 소프트웨어 리버싱
- 위와 같은 순서로 보통 학습이 이루어짐
- 보통 대회에서는 관리자 권한의 셸을 실행시키는 것이 목적

Stack Frame

```
#include <stdio.h>

int main() {
    char buf[0x30];
    scanf("%s", buf);
    return 0;
}
```



Disassemble it!

```
pwndbg> disassemble main
Dump of assembler code for function main:
0x0000000000001149 <+0>:    endbr64
0x000000000000114d <+4>:    push    rbp
0x000000000000114e <+5>:    mov     rbp, rsp
0x0000000000001151 <+8>:    sub     rsp, 0x30
0x0000000000001155 <+12>:   lea     rax, [rbp-0x30]
0x0000000000001159 <+16>:   mov     rsi, rax
0x000000000000115c <+19>:   lea     rax, [rip+0xea1]      # 0x2004
0x0000000000001163 <+26>:   mov     rdi, rax
0x0000000000001166 <+29>:   mov     eax, 0x0
0x000000000000116b <+34>:   call    0x1050 <__isoc99_scanf@plt>
0x0000000000001170 <+39>:   mov     eax, 0x0
0x0000000000001175 <+44>:   leave
0x0000000000001176 <+45>:   ret
End of assembler dump.
```

Function Prologue, Eplilogue

```
0x0000000000000114d <+4>:  push  rbp  
0x0000000000000114e <+5>:  mov   rbp, rsp
```

함수 프로로그

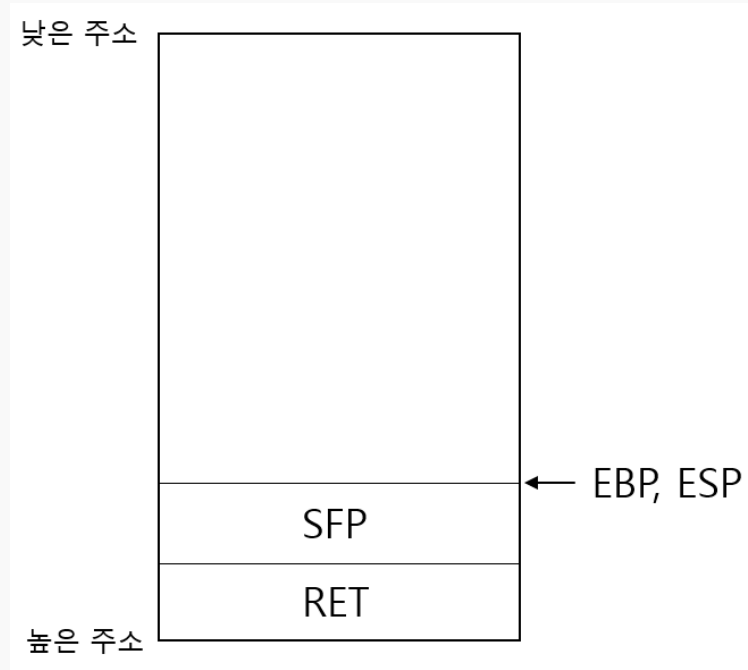
```
0x00000000000001175 <+44>:  leave  
0x00000000000001176 <+45>:  ret
```

함수 에필로그



Function Prologue

```
push    rbp
mov     rbp, rsp
```

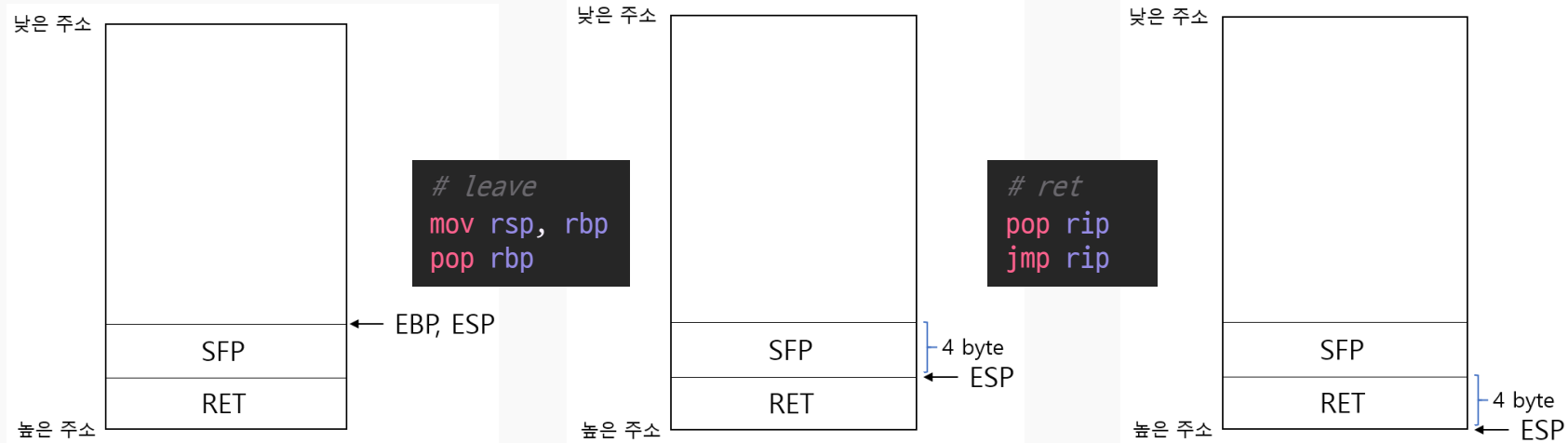


Function Epilogue

```
0x00000000000001175 <+44>:  leave  
0x00000000000001176 <+45>:  ret
```

```
# leave  
mov rsp, rbp  
pop rbp  
  
# ret  
pop rip  
jmp rip
```

Function Epilogue



Buffer OverFlow, BOF

```
#include <stdio.h>

int main() {
    char buf[0x30];
    scanf("%s", buf);
    return 0;
}
```

- 해당 프로그램에서 scanf 함수는
입력길이의 제한이 없음
- 저장하는 buf 변수의 크기를
넘어가게 입력이 가능

→ Buffer OverFlow!

Buffer OverFlow, BOF

- 스택은 낮은주소 부터 높은주소로 쓰여짐
- buf 의 크기를 넘겨 입력하면, SFP 와 RET 주소를 덮을 수 있음

→ 함수 실행의 흐름을 바꿀 수 있음

Return Oriented Programming, ROP

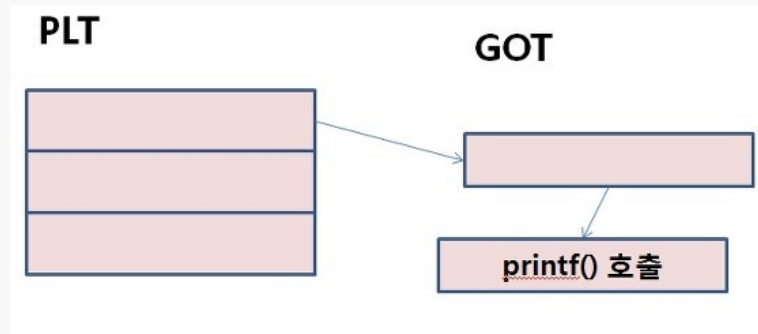
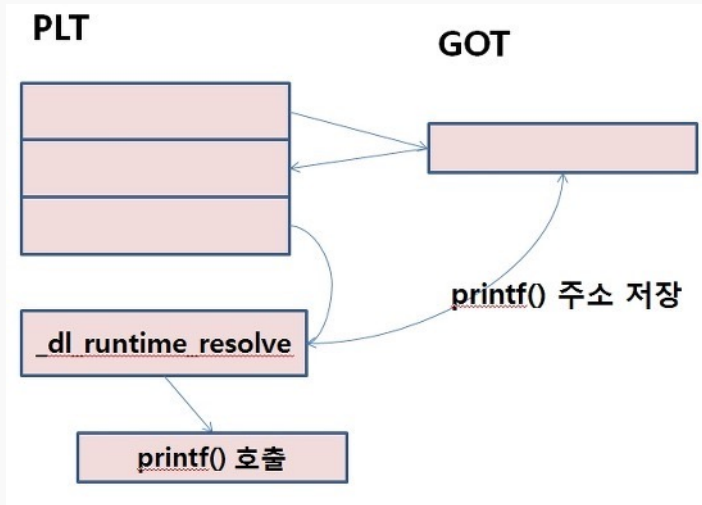
- 마치 프로그래밍 하듯이 리턴 주소를 조작해 함수를 실행하는 기법
- BOF 를 일으켜 리턴 주소 이후를 덮을 수 있을 때 사용된다.

사전지식: GOT 와 PLT, ASLR, Shared Library, RTL

GOT, PLT

- GOT (Global Offset Table) → 호출하는 함수의 실제 주소를 구하는 코드 (PLT+6) 의 주소를 담고 있다가, 함수가 최초로 호출되면, 함수의 실제주소를 담는 테이블
- PLT (Procedure Linkage Tabel) → 파일 내부가 아니라, 다른 라이브러리에 함수를 호출할 때 연결시켜주는 테이블

GOT, PLT



Shared Library와 ASLR

- 공유 라이브러리
- 리눅스에는 **.so** 파일, 윈도우에는 **.dll** 파일
- 프로그램이 실행될 때마다 가상 주소공간에 올라가는 스택, 힙, 공유 라이브러리의 위치가 랜덤으로 변함

Shared Library와 ASLR

```
pwn@citrusinesis:~/binary$ ./a.out
[stack] (int a=10) a in 0x7ffe8e77cd84
[heap] (int *p=malloc(0x20)) the value of p is 0x559682a052a0
[shared library] the address of 'puts' function is 0x7f3b7e90ded0
pwn@citrusinesis:~/binary$ ./a.out
[stack] (int a=10) a in 0x7fffbb84a5b4
[heap] (int *p=malloc(0x20)) the value of p is 0x55c270ec62a0
[shared library] the address of 'puts' function is 0x7fb5bf0b1ed0
pwn@citrusinesis:~/binary$ ./a.out
[stack] (int a=10) a in 0x7fffc2d1eba4
[heap] (int *p=malloc(0x20)) the value of p is 0x5634fa0812a0
[shared library] the address of 'puts' function is 0x7ff2b6384ed0
```

Return To Libc, RTL

1. 화면에 출력해주는 함수(puts,write,printf) 등으로 리턴해 바이너리내에 존재하는 함수의 실제 주소 (got) 출력
2. $\text{got} - \text{offset} \rightarrow$ 라이브러리가 로딩된 주소(libc_base)
3. 그리고 libc_base 에 원하는 함수의 offset 을 더해 실제 주소를 구해 RTL

Pwned!


```
[*] Switching to interactive mode
\xa0\xec\xd6\xf7F\x83\x040\xfe\xd1\xf7
$ id
[DEBUG] Sent 0x3 bytes:
      b'id\n'
[DEBUG] Received 0x33 bytes:
      b'uid=0(root) gid=0(root) groups=0(root),999(docker)\n'
uid=0(root) gid=0(root) groups=0(root),999(docker)
$ █
```

이보다 훨씬 많은 기법들

- Return To CSU
- STACK-PIVOTING
- Format String Bug
- Out Of Bound

등등 ...





감사합니다

“

```
function filterStudies(studies, filterByOrg = false, filterByYear = false) {
  return studies.filter(study => {
    if (filterByOrg) {
      return study.organizationalUnit !== "Other"
    }
    if (filterByYear) {
      return study.startYear < 2010
    }
    return true
  })
}
```