# HW 1: Buffer Overflow

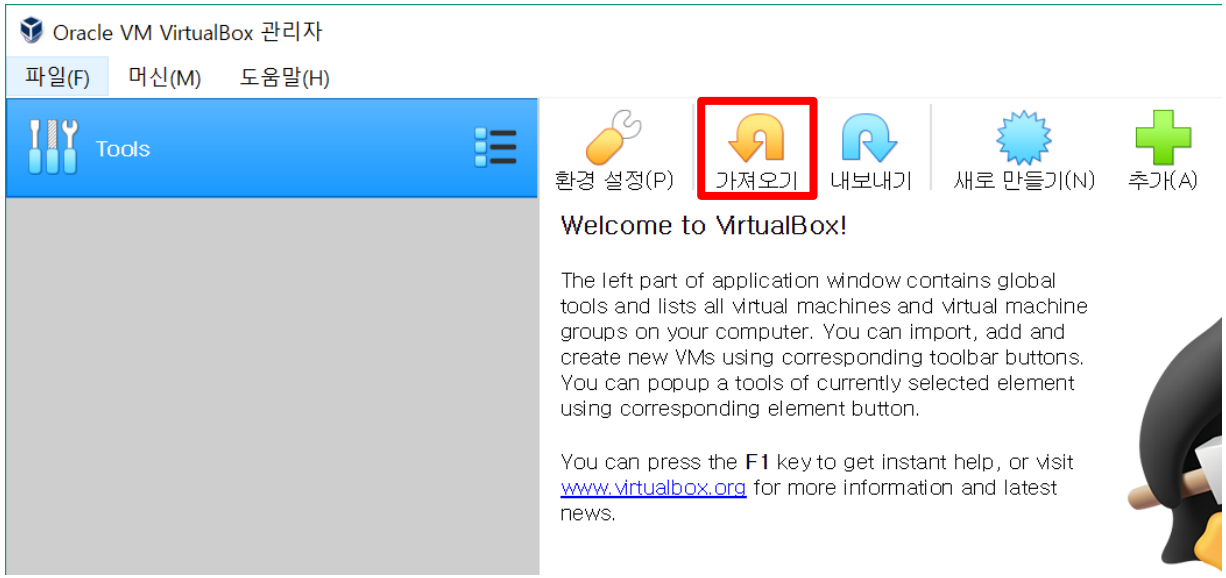
# Index

- 실습 환경
- 개인 과제 목록
- Buffer Overflow
- 과제
- 제출

# 개인 과제 및 실습 – VM 설정

- SEED Labs (http://www.cis.syr.edu/~wedu/seed/)
  - SEEDUbuntu-16.04 수정본에서 진행

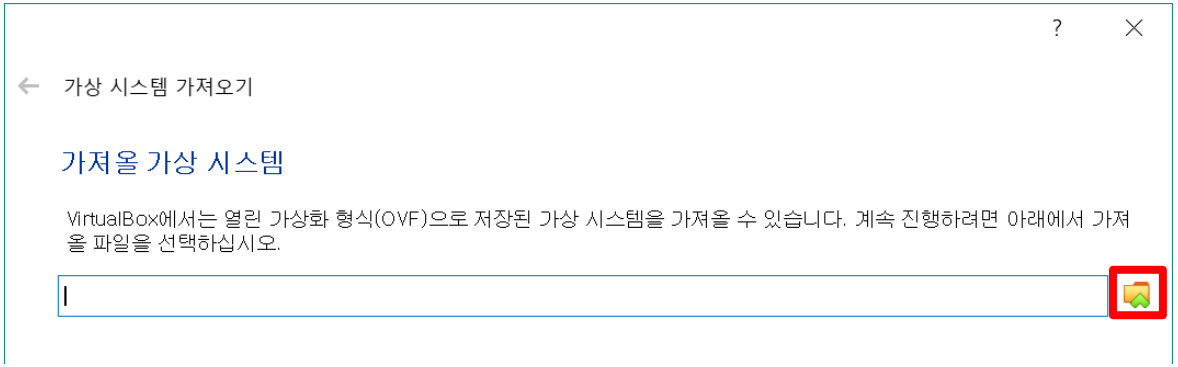
- Setup
  1. VM 다운로드 : **연구실 홈페이지에 링크 게재**
  2. 다운로드 받은 파일의 압축 해제
  3. “VirtualBox” 설치 및 실행
  4. “가져오기”

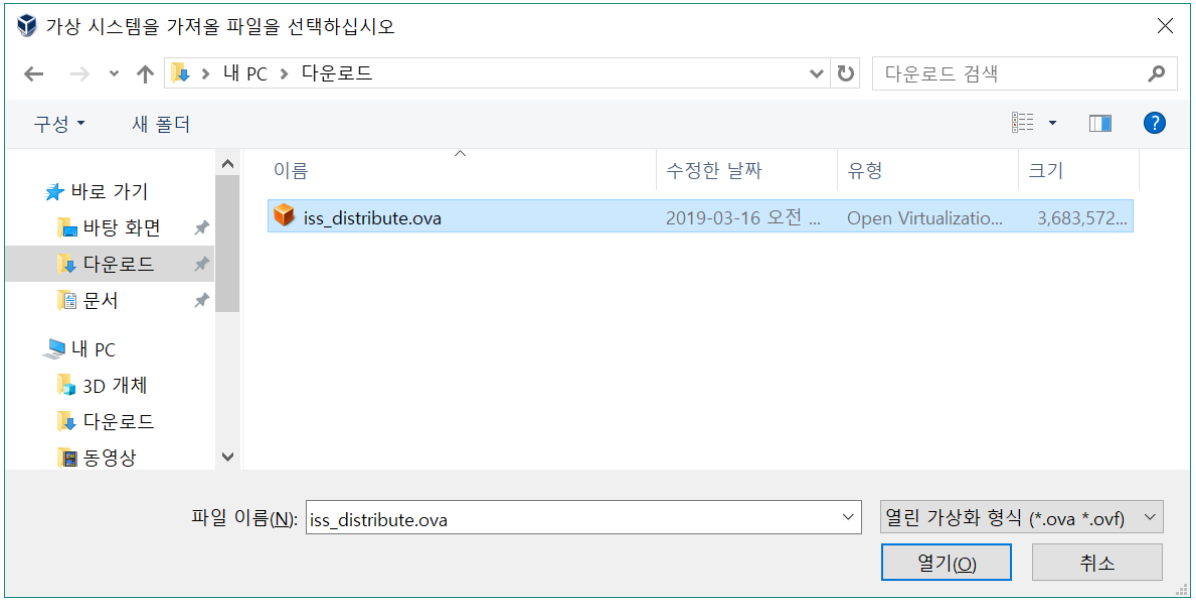


# 개인 과제 및 실습 – VM 설정

❖ Setup

5. 다운로드 받은 파일 선택 > “다음”



# 개인 과제 및 실습 – VM 설정

❖ Setup

6. “가져오기”


# 개인 과제 및 실습 – VM 설정

❖ Setup

7. “시작”으로 VM 전원 ON

8. ID: seed / PW: dees

# HW1

- 버퍼 오버플로우 취약점 실습
  - 버퍼 오버플로우 취약점 실습 및 분석
  - HW1_BOF 폴더

- 과제
  1. BOF 취약점이 존재하는 소스코드와 실행파일 제공 (*bof.c*, *bof*)
  2. 소스코드 상에서 프로그램의 기능을 변경하지 않으면서 취약점을 보완할 수 있는 방안을 제시할 것
  3. 제공된 exploit 코드(*exploit.txt*)를 참고하여 권한 상승
  4. exploit 코드의 체계적인 분석

- 제출
  - 과제 및 실습 수행 내용을 보고서로 제출
  - 제출기간 : 3월 20일 ~ 4월 3일

# HW2

❖ 암호화 관련 실습
- 간단한 암복호화 수행 및 MD5의 충돌 실습
- HW2_Crypto 폴더

❖ 과제
1. 치환 암호 기법으로 작성된 파일(*ciphertext.txt*)을 복호화
2. 복호화된 *ciphertext.txt*에서 확인할 수 있는 key로 암호화된 이미지를 복호화
3. MD5 hash 생성 및 충돌을 확인

❖ 제출
- 과제 및 실습 수행 내용을 보고서로 제출
- 단, 복호화한 이미지가 반드시 보고서에 포함되어야 함
- 제출기간 : 4월 22일 ~ 5월 8일

# HW3

❖ SQL Injection 실습
- DB대상 공격을 수행하여 정보 수정 및 관리자 계정 획득
- HW3_SQLInjection 폴더

❖ 과제 내용
1. PW를 모르는 상태로 관리자 계정 획득
2. 내 계정의 봉급 정보 수정
3. 타인의 정보 수정
4. SQL Injection 취약점의 개선 방안 분석

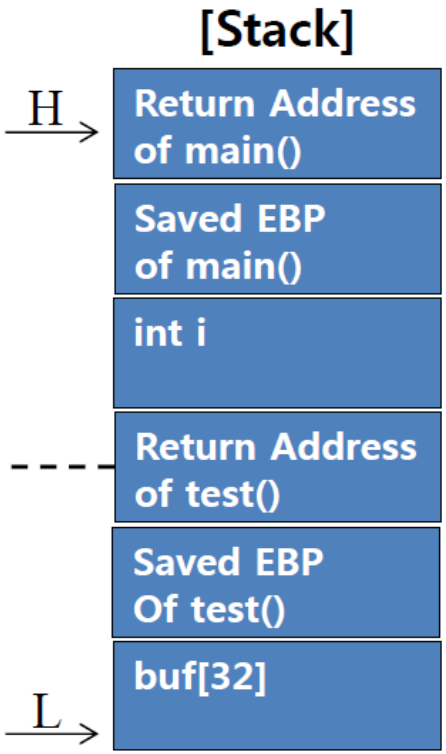
❖ 제출
- 과제 및 실습 수행 내용을 보고서로 제출
- 제출기간 : 5월 29일 ~ 6월 5일

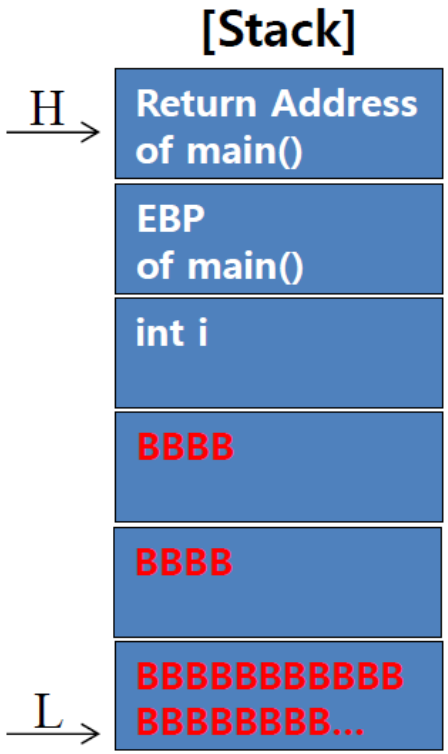
# Buffer Overflow

- BOF(Buffer Overflow)
  - 메모리 공간을 초과한 입력을 허용하여 발생하는 취약점
    - 이 취약점을 악용해 임의 파일을 읽거나 쉘(/bin/sh)을 띄우는 것을 목표로 함
    - exploit으로 BOF를 발생시킴

```
void test()
{
    char buf[32];
    scanf("%s",buf);
}

int main()
{
    int i=0x41414141;
    test();
    return 0;
}
```

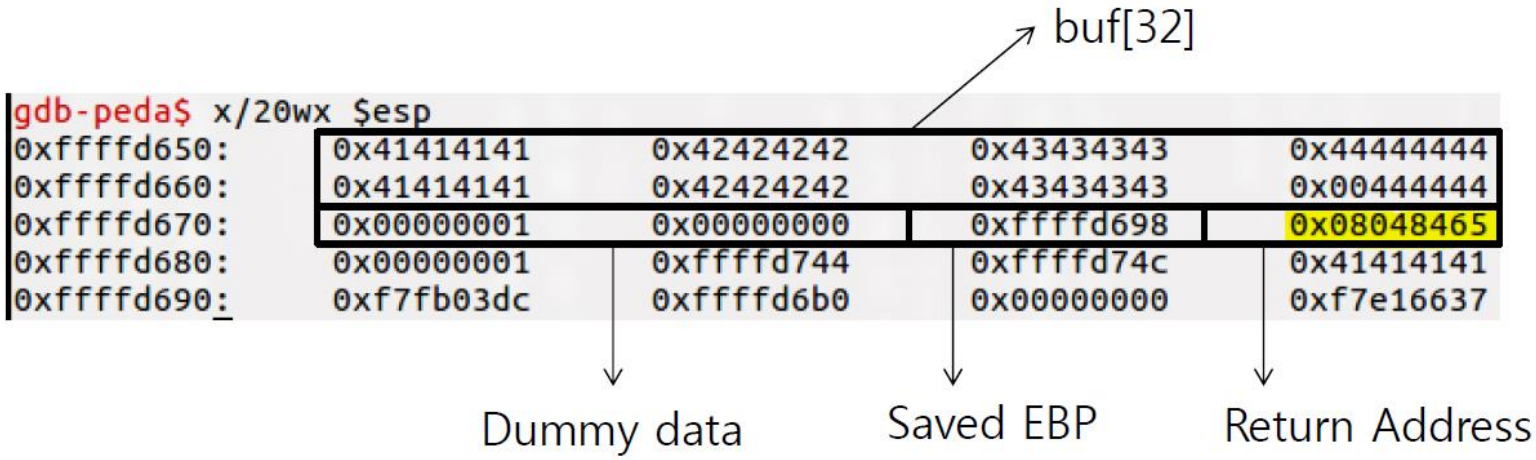


# Buffer Overflow

- BOF(Buffer Overflow)
  - 메모리 분석(gdb)

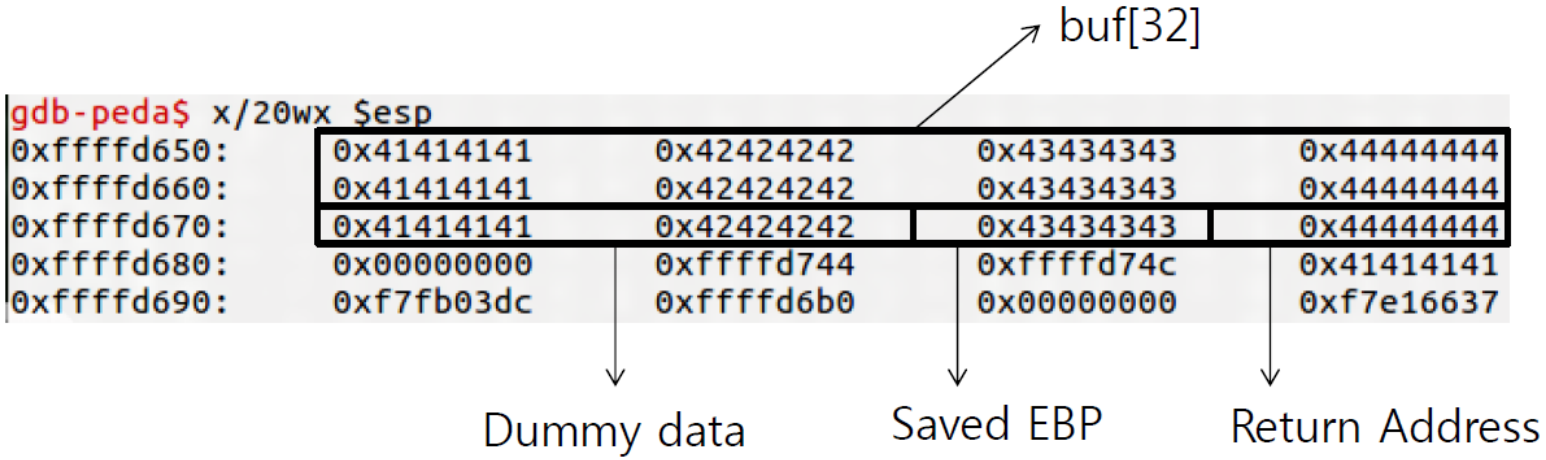
test() 함수의 스택프레임
31바이트의 임의 값을 넣었을 경우



# Buffer Overflow

- BOF(Buffer Overflow)
  - 메모리 분석(gdb)

test() 함수의 스택프레임
48바이트의 임의 값을 넣었을 경우




잘못된 주소지를 참조하는 것을 확인(Segmentation Fault)

```
root@ubuntu:~/Desktop# ./test
AAAABBBBCCCCDDDDAAAABBBBCCCCDDDDAAAABBBBCCCCDDDD
Segmentation fault (core dumped)
```

# Buffer Overflow

❖ Exploit ?

- 컴퓨터의 소프트웨어나 하드웨어 및 컴퓨터 관련 전자 제품의 **버그, 보안 취약점** 등 설계상 결함을 이용하여 공격자의 의도된 동작을 수행하도록 만들어진 **절차**나 일련의 **명령**, **스크립트**, **프로그램** 또는 **특정한 데이터 조각**
- 이러한 것들을 사용한 공격 행위를 의미함
- 본 과제에서는 완성된 exploit을 제공함

# GDB

- GDB(GNU Debugger)
  - debugger : 프로그램을 테스트하고 디버그하는 일종의 프로그램
  - 프로그램의 실행흐름 추적
  - 메모리 및 변수의 값 확인 및 변경 가능

```
GNU gdb 5.2.1
Copyright 2002 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i686-pc-mingw32".
(gdb) help
List of classes of commands:

aliases -- Aliases of other commands
breakpoints -- Making program stop at certain points
data -- Examining data
files -- Specifying and examining files
internals -- Maintenance commands
obscure -- Obscure features
running -- Running the program
stack -- Examining the stack
status -- Status inquiries
support -- Support facilities
tracepoints -- Tracing of program execution without stopping the program
user-defined -- User-defined commands

Type "help" followed by a class name for a list of commands in that class.
Type "help" followed by command name for full documentation.
Command name abbreviations are allowed if unambiguous.
(gdb) q
```

# GDB

## ❖ GDB 커맨드

- gdb 실행 : gdb [file]

```
[03/19/19]seed@VM:~/.../HW1_BOF$ gdb bof
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.04) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/lice
Type "apropos word" to search for commands related to "word"...
Reading symbols from bof...(no debugging symbols found)...done.
gdb-peda$
```

- Code disassemble : disas [function name]
  - e.g) disas main

    disas func

```
gdb-peda$ disas main
Dump of assembler code for function main:
   0x080484ee <+0>:     lea    ecx,[esp+0x4]
   0x080484f2 <+4>:     and    esp,0xfffffff0
   0x080484f5 <+7>:     push   DWORD PTR [ecx-0x4]
   0x080484f8 <+10>:    push   ebp
   0x080484f9 <+11>:    mov    ebp,esp
   0x080484fb <+13>:    push   ecx
   0x080484fc <+14>:    sub    esp,0x4
   0x080484ff <+17>:    sub    esp,0xc
   0x08048502 <+20>:    push   0xdeadbeef
   0x08048507 <+25>:    call   0x804849b <func>
   0x0804850c <+30>:    add    esp,0x10
   0x0804850f <+33>:    mov    eax,0x0
   0x08048514 <+38>:    mov    ecx,DWORD PTR [ebp-0x4]
   0x08048517 <+41>:    leave
   0x08048518 <+42>:    lea    esp,[ecx-0x4]
   0x0804851b <+45>:    ret
End of assembler dump.
gdb-peda$
```

# GDB

- 그 밖의 GDB 커맨드
  - Breakpoint 설정 : b *[address]
    - e.g) b *0x80010203

      b *main+4
  - 프로그램 실행(run) : r
  - 중단된 프로그램 실행 : c
    - breakpoint로 중단된 시점부터 실행 가능
  - 어셈블리 단위 실행 : ni

# 과제

❖ 버퍼 오버플로우 취약점

① HW1_BOF 폴더에 제공되는 프로그램에서 BOF를 일으킬 것

- 버퍼오버플로우 취약점을 가지는 프로그램(*bof*)과 소스코드(*bof.c*)를 제공

1) 터미널에서 다음 exploit을 입력하여 BOF를 발생시킬 수 있음

- (python –c 'print "a"*48+"\xbe\xba\xfe\xca"'; cat) | ./bof

```
[03/15/19]seed@VM:~/.../HW1_BOF$ (python -c 'print "a"*48 + "\xbe\xba\xfe\xca"';cat)|./bof
whoami
root
```

- exploit을 통하여 root 권한의 shell을 획득
- whoami 명령어로 root로 로그인되었음을 확인

- 취약점이 패치된 프로그램(*bof_notvul*)

2) *bof_notvul*에 exploit 입력 시 작동되지 않는 것을 확인

```
[03/15/19]seed@VM:~/.../HW1_BOF$ (python -c 'print "a"*48 + "\xbe\xba\xfe\xca"';cat)|./bof_notvul
overflow me : Nah..
whoami
[03/15/19]seed@VM:~/.../HW1_BOF$
```

# 과제

❖ 버퍼 오버플로우 취약점

② 10페이지에 제공된 exploit을 분석

- *bof.c* 코드 분석
- gdb를 통해 *bof*의 메모리 구조 분석
- **exploit이 어떻게 작동하는지를 분석**
- hint) gdb로 'func' 함수 분석

③ 취약점의 보완

- *bof.c*에서 **취약점을 가지는 부분을 찾아 보완할 것**
- 본래 기능을 상실하지 않으며 실행이 가능할 것
- cf) 컴파일은 다음의 명령어로 가능함
  - gcc –fno-stack-protector –z execstack –o bof_new bof.c

# 제출

- 보고서는 다음을 포함하여야 함
  - 과제 ①~③의 수행 과정 및 결과
  - 수행 과정 및 분석에 대한 논리적인 서술

- 제출 기간: 3월 18일 ~ 4월 3일
- 수업시간에 제출하거나 미디어센터 505호로 방문 제출
  (부재 시 504호로 제출)

# 문의사항

- 조교 이름 : 정재민
- 연락처 : s17orlax@gmail.com

# Thank you!

# Q & A