Dreamhack Return Address Overwrite Writeup

```c
void init() {
  setvbuf(stdin, 0, 2, 0);
  setvbuf(stdout, 0, 2, 0);
}

void get_shell() {
  char *cmd = "/bin/sh";
  char *args[] = {cmd, NULL};

  execve(cmd, args, NULL);
}

int main() {
  char buf[0x28];

  init();

  printf("Input: ");
  scanf("%s", buf);

  return 0;
}
```

제공되는 rao.c 소스코드이다.

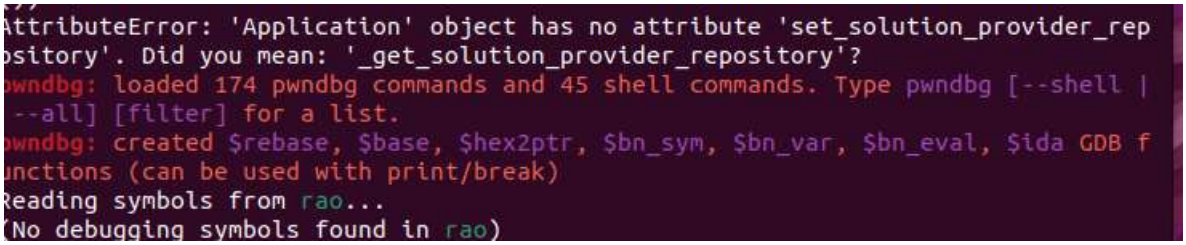
취약점은 scanf("%s", buf)에 있다. %s를 사용했기 때문에 버퍼 오버플로우가 발생할 수 있다.

이 예제에서는 크기가 0x28인 버퍼에 scanf("%s", buf)로 입력을 받으므로 입력을 길게 준다면 버퍼 오버플로우를 발생시켜서 main함수의 반환 주소를 덮을 수 있다.

```
yuna@yuna-virtual-machine:~$ gcc -o rao rao.c -fno-stack-protector -no-pie
yuna@yuna-virtual-machine:~$ ./rao
Input: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Segmentation fault (core dumped)
```

이렇게 버퍼 오버플로우를 발생시키면 Segmentation fault라는 에러가 출력된다. 이는 프로그램이 잘못된 메모리 주소에 접근했다는 의미이며, 프로그램에 버그가 발생했다는 신호이다.

뒤의 (core dumped)는 코어 파일을 생성했다는 것으로, 프로그램이 비정상 종료됐을 때, 디버깅을 돕기 위해 운영체제가 생성해주는 것이다.



이제 디버거에서 생성된 코어 파일을 열어보려 했는데 디버깅 심볼이 없다면서 제대로 실행이 되지를 않는다...