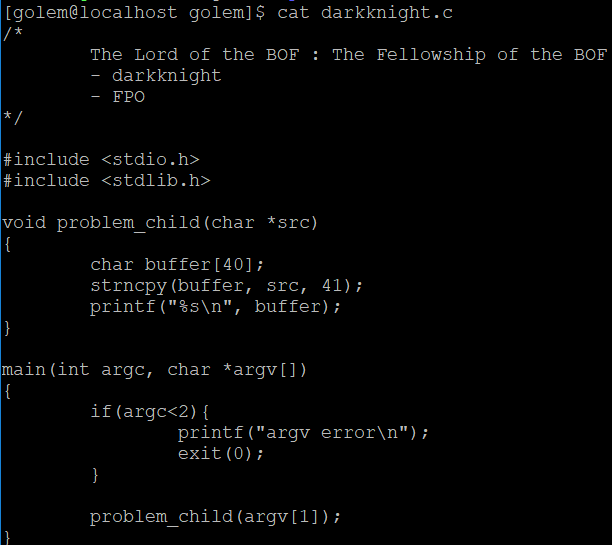
|  |
| --- |
| LOB darkknight |
| 동아리마크  2018. 02. 28 (wed)  Dong-a .univ  COLONY  Young jin kim |

목차

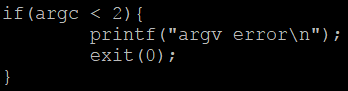
1. 소스코드 이해
2. 소스코드 분석
3. 공격
4. **소스코드 이해**



[ golem.c의 소스코드 ]

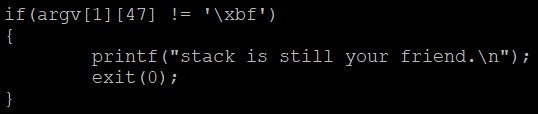
|  |  |
| --- | --- |
| Problem\_child함수 정의  Char형 buffer 40byte 선언  Src에서 41byte만큼 buffer에 복사  Buffer 출력 | main함수 정의  인자값이 1개 이하이면 프로그램 종료  Problem\_child 함수 실행  FPO(Frame Pointer Overfow) : sfp 조작기법  -FPO를 통해 원하는곳으로 Return이 가능하다 |

1. **소스코드 분석**



[ 조건 1 ]

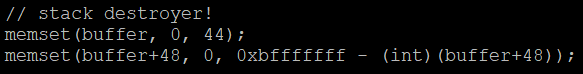
첫번째, 인자는 2개 이상 사용한다



[ 조건 2 ]

첫번째, 첫번재 인자의 48번째 데이터는 \xbf가 들어가야한다.

두번째, 우리가 ret하는 지점의 메모리값이 강제 된다.



[ 조건 3 ]

첫번째, buffer이후의 ret 4byte를 제외한 모든 메모리를 초기화한다.

두번째, 쉘코드는 초기화 되는 부분에는 들어가기 힘들어 보인다.

1. **소스코드 분석**



[ 조건 4 ]

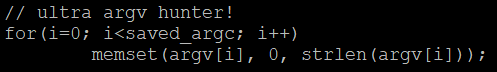
첫번째, argv[1]의 길이가 48보다 크면 프로그램 종료



[ 조건 5 ]

첫번째, buffer변수 초기화

두번째, buffer에도 쉘코드를 넣는 것은 불가능하다.



[ 조건 6 ]

첫번째, 인자를 순차적으로 초기화 한다.

두번째, 이름에 쉘코드 삽입도 불가능해졌다.

1. **공격**



[ LD\_PRELOAD 1 ]

첫번째, LD\_PRELOAD는 초기화하는 부분의 메모리값과 다른 부분에 위치한다.

두번째, 이 환경변수는 라이브러리를 사용하기위해 사용하는 변수다.

세번째, kyj.c라는 파일을 생성한다.



[ LD\_PRELOAD 2 ]

첫번째, kyj.c 파일을 gcc -shared -fPIC -o 로 컴파일을 진행하였다.

두번째, 컴파일하는 프로그램명에는 쉘코드가 적재되어 있다.



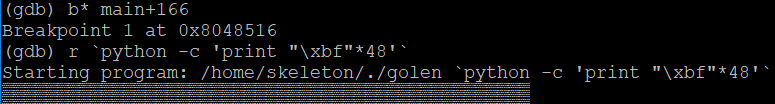
[ LD\_PRELOAD 3 ]

첫번째, 쉘코드 적재가 잘 되었다.

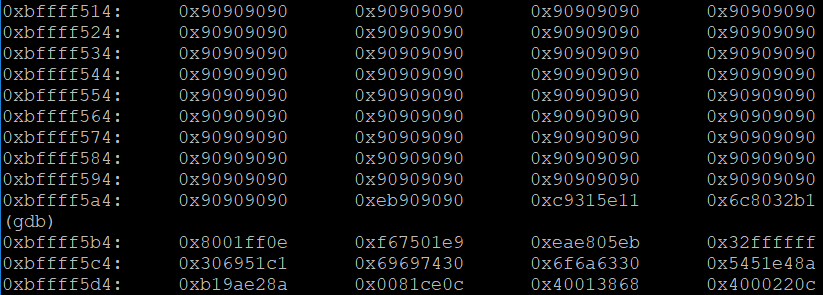
1. **공격**



[ gdb 분석 1 ]



[ gdb 분석 2 ]



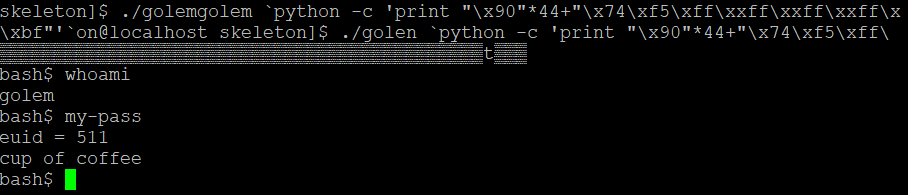
[ gdb 분석 3 ]

첫번째, 놉뒤에 무언가가 들어가 있는 것을 확인하였다.

두번째, 무언가는 쉘코드

세번째, 0xbffff574를 ret로 공격을 시작해보겠다.

**3.공격**



[익스플로잇]

Pw : cup of coffee