|  |  |  |  |
| --- | --- | --- | --- |
| LOB Gremlin | | | |
| COLONY | 동아대학교 | 김영진 | 2017.01.05 |
| Gremlin 문제풀이 시작하겠습니다.    홈디렉터리 안에있는 코볼트 코드입니다.  역시 gate와 마찬가지로 argv[1]에서 버퍼로 문자열 복사할 때 버퍼오버플로우가 발생함을 알 수 있습니다.  차이점은 버퍼의 크기가 작다. 이점이네요  앞서 썻던 25바이트의 쉘코드는 사용하기 힘들겠네요. 아니면 아에 다른 방법을 사용해야 할 것같습니다,.  Gcc 버전을 확인했습니다. 2.91버전은 더미가 업고 2.96보다 상위버전은 더미가 생깁니다.    역시 권한이 없기 때문에 디버깅을 위해 카피를 합니다.    Gdb로 디버깅을 시작해보겠습니다.    전체적인 구조는 gate와 비슷하고 버퍼크기 차이만 있네요.  버퍼크기는 16바이트가 할당됬네요    [ 버퍼(16) ] + [ ebp(4) ] + [ ret(4) ] 구조에서  [ 20byte ”A” ] + [ NOP(20) ] + [ ShellCode ] 로 페이로드를 짰다  ret부분에서 놉 슬레이드가 일어나서 쉘코드가 실행되는 구조이다.  메모리 맵 확인!    잘 들어갔네요.  위의 페이로드로 공격을 해 보았는데 세그먼트 폴트가 떳다.  아무래도 ret값에는 주소를 그대로 넣어주고 ret뒤쪽으로 값을 넣어보았다.  페이로드는 A(20byte)+주소+놉(100)+ShellCode이다  쉘코드가 시작되는 주소만 찾고 바로 공격을 시도해보겠다.    우리가 넣을 쉘코드의 시작지점은 43434343부분의 주소인 0xbffffaf4이다.  오류가 뜹니다.  코어파일이 생성되도록 해서 코어파일을 분석해봅시다.  Ulimit -c unlimited 명령으로 코어파일 길이 제한 해제  코어파일을 디버깅 해 보았다    다시 주소를 확인 해 보니 bffffae0으로 나왔다  주소를 고쳐서 다시 익스플로잇!    성공!  교훈 : 쉘코드는 꼭 버퍼에 넣을 필요는 없다. | | | |