Crack me 문제 보고서

|  |  |
| --- | --- |
| 문제명 | uBc crackme 01 |
| 파일명 | uBc01.exe |
| 풀이자 | 서치원(2011003155) |
| 풀이 기간 | 2016/01/17 ~ 2017/01/23 |
| 문제 내용 | \* Name과 serial의 조합에서 name이 CodeEngn일 때의 serial값을 구하라. |

문제 풀이:

문제 파일을 실행 시 창 하나가 뜨게됩니다.



name과 그에 맞는 serial을 찾는 조건인듯한데 홈페이지의 문제 내용에선 name이 "CodeEngn"일때의 serial 값을 찾는 것입니다.



ollydbg에서 시작시의 화면입니다. 코드의 진행을 알기 힘들게 되어 있는데 PEiD로 검사해본 결과 Delphi 언어로 작성됨을 확인하였습니다. 일단 참조하는 텍스트 검색으로 정답을 출력하는 위치를 검색해봅니다.



참조하는 텍스트 중 정답임을 알리는 문자열이 있습니다. 저 주소로 이동합니다.



주소로 이동한 결과 특정 메소드로 이동하였습니다. name과 serial에 값을 넣고 실행해보면 실제로 이 메소드에 도달하게 됩니다.

메소드를 살펴모면 458837에서 EAX가 45B844 주소에서의 DWORD의 값과 다르면 오답으로 처리하게 됩니다. 이 주소에는 0x60, 0x61이 저장되어 있어 EAX는 0x00006160이 되어야 합니다. EAX는 초기에 0으로 초기화 되고 3개의 메소드를 통해 변하게 됩니다. 첫 번째 메소드 458760은 인자로 458874 주소의 값과 EAX값을 받습니다.



첫 번째 메소드의 실행 후의 레지스터 모습입니다. EDX에 입력한 name값이 저장되어 있음을 확인할 수 있습니다.

첫 번째 메소드의 실행 후에는 EDX, EAX에 특정 값을 저장하는데 그 중, EAX에는



에서 가리키는 0x01F4A60C 값을 읽게됩니다.



계속 진행해서 두 번째 메소드를 읽은 후의 레지스터 값입니다. EAX에 serial값이 저장되어 있습니다. 이 상태에선 세 번째 메소드의 결과가 EAX 값을 결정하게 될 것이라 보여집니다.



세 번째 메소드의 루틴입니다. 내부에서 또 다시 메소드 call을 하는데 그 당시의 레지스터 상황입니다. EAX에 serial값이 들어가 있습니다.



내부에서 Call 된 메소드로 들어온 화면입니다. break 건 루프에서 serial 값을 차례대로 읽어 계산을 하게 되는데 결과적으로 단순히 숫자의 문자열을 10진수에서 16진수로 변경하는 루틴입니다. 즉, name은 serial의 결정에 영향을 끼치지 않고 serial은 16진수로 0x6160인 10진수 값 24928이 정답입니다.



정답 serial을 입력해 나온 결과입니다.