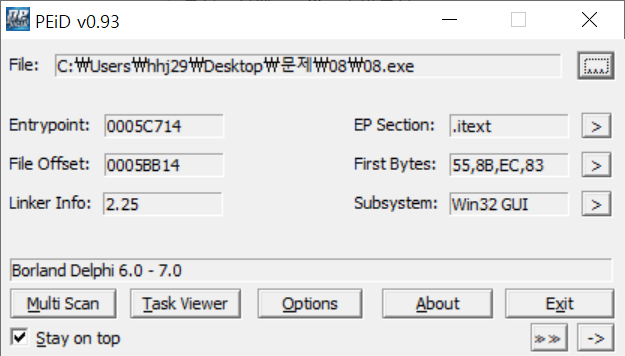
[CodeEngn] 8번

1. 문제 확인



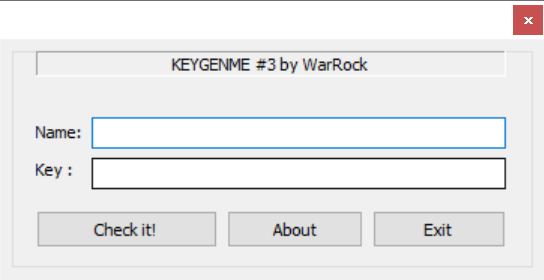
2. 과정

해당 문제를 열기에 앞서 PEiD로 패킹 여부를 확인해보았다.



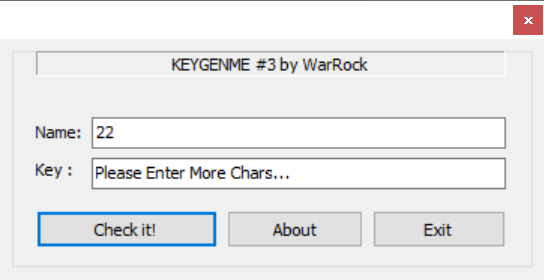
패킹 되어있지는 않고 델파이로 작성된 것을 알 수 있다.

프로그램을 실행해보니



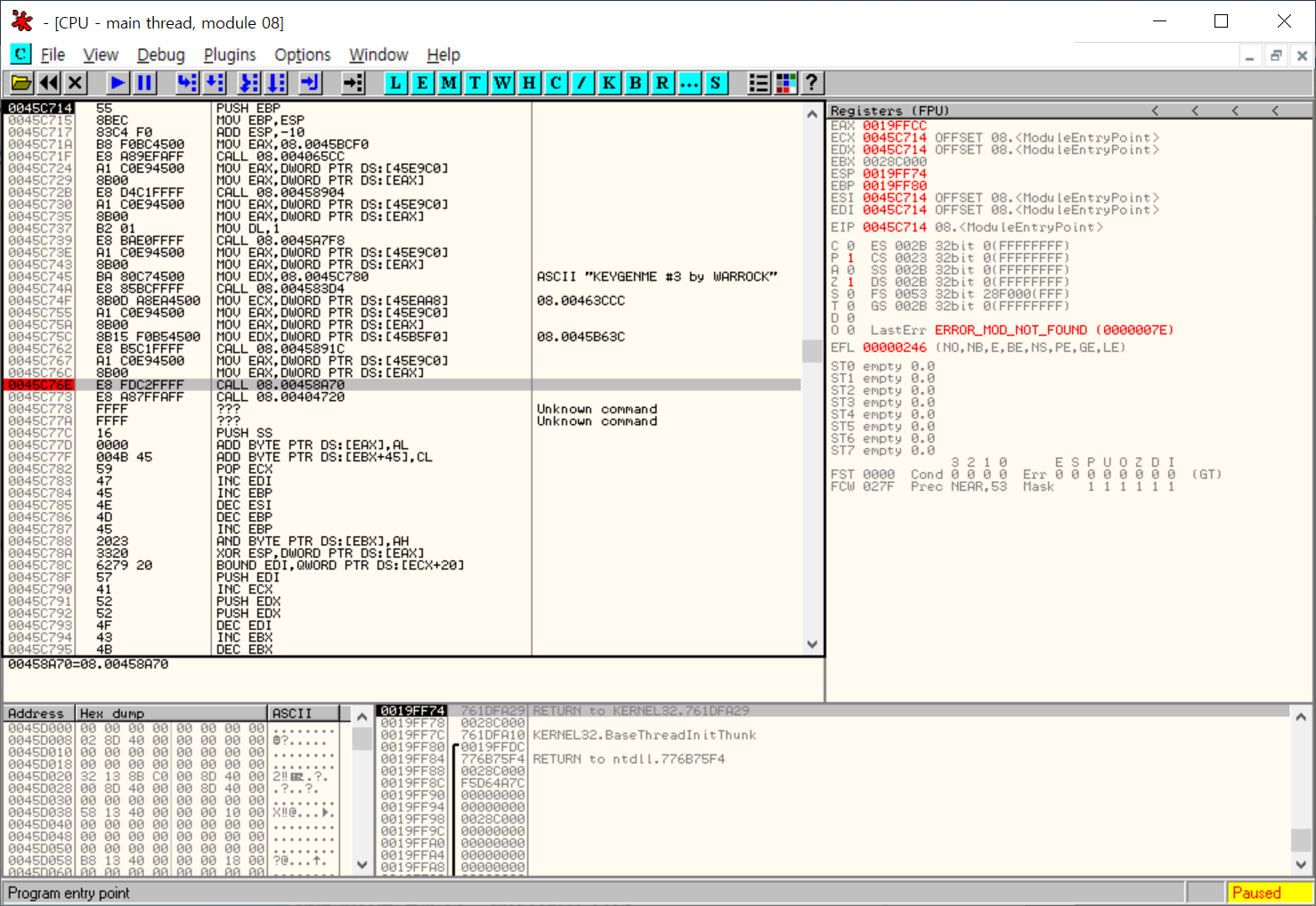
네임과 키를 작성하는 창이 뜨는데 여기서 key의 값은 5D88-53B4-52A87D27-1D0D-5B09 문제에서 미리 주어졌다. name은 두자리라고 힌트에 적혀 있다.

아무 이름 2글자와 key를 입력해보니

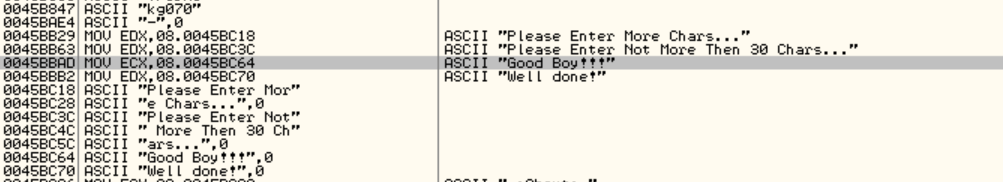


다음과 같은 더 입력하라는 문자열이 나온다.

올리디버거로 열어보았다.



아까 뜬 문자열이 어디에 나오는지 찾아보기 위해 text strings를 검색해보았다.



“Please enter more chars”와 함께 성공문자열도 찾은 것 같다.

일단 이 부분을 확인하기 위해 breakpoint를 걸어 두고

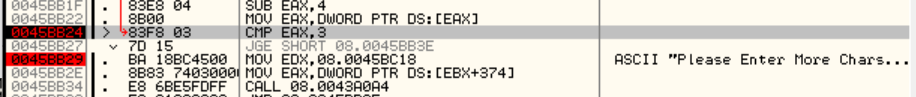
다시 돌아가 한줄씩 디버깅을 해보았다.

Call 부분에서 문제창이 뜨면서 입력 값을 받게 된다.



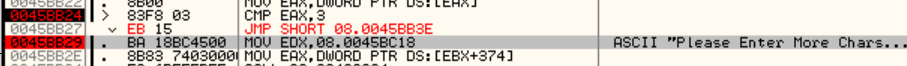
Name에 ab를, 시리얼에는 5D88-53B4-52A87D27-1D0D-5B09를 입력했다.

엔터를 누르니까 아까 breakpoint를 걸어 둔 문자열로 이동했다.

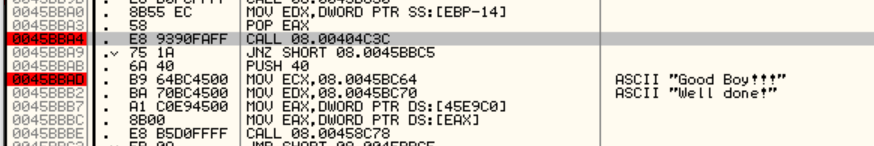


여기서 위 코드를 보면 CMP 비교문이 있는데 EAX 값이 3이면 넘겨주고 아니면 “Please Enter More…” 이라는 문자열이 출력되는 것으로 보인다. name으로 받는 값은 2글자라고 했는데 왜 3과 비교하는지 잘 모르겠다..

그래서 그냥 JGE문을 JMP로 변경해주고 계속 진행했다.

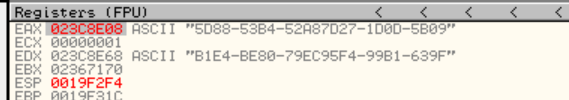


코드에서 045BBA4의 CALL 함수를 지나치면

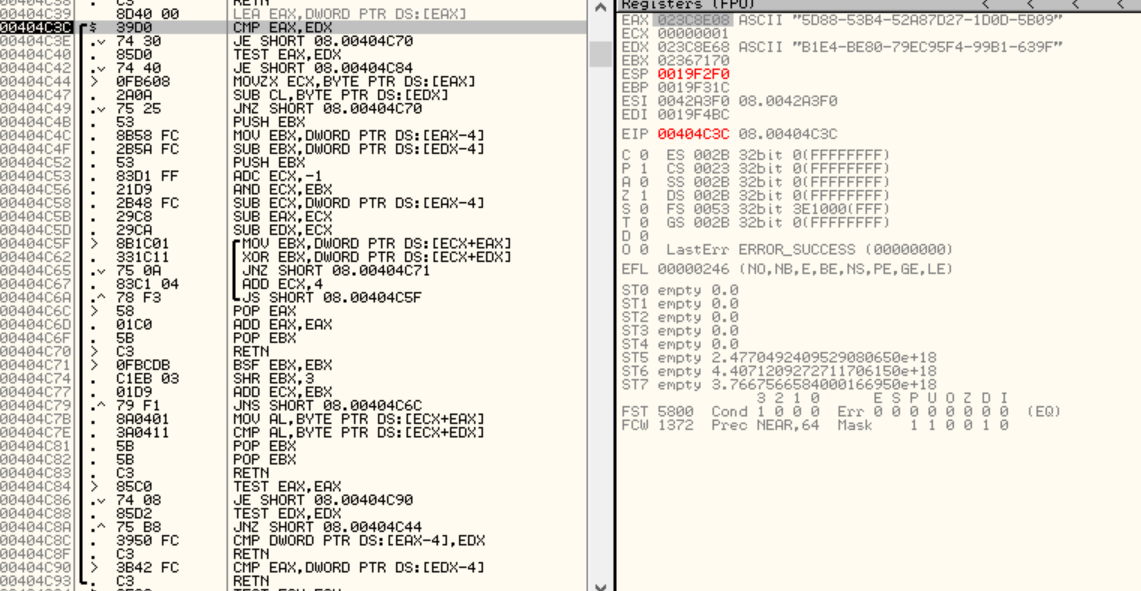


레지스터에 입력한 값이 저장되는 것을 확인할 수 있다.

EAX에는 내가 시리얼 값으로 입력한 정보가 들어있는데 EDX에는 시리얼 형식인 다른 값이 들어있었다.

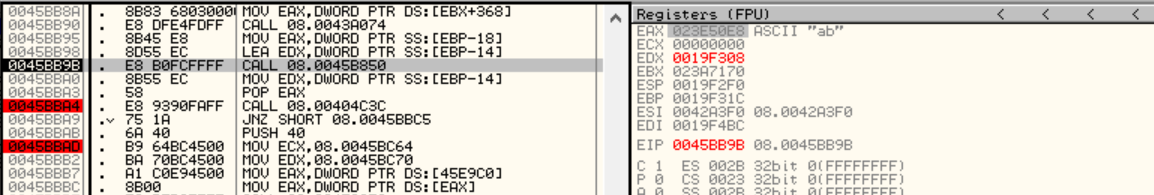


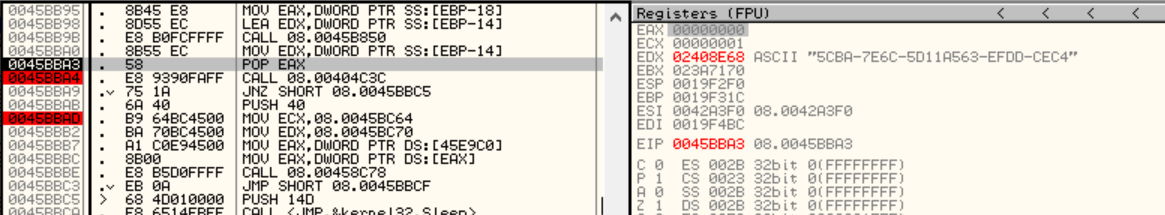
F7을 눌러 Call 함수를 자세히 확인해보았다.



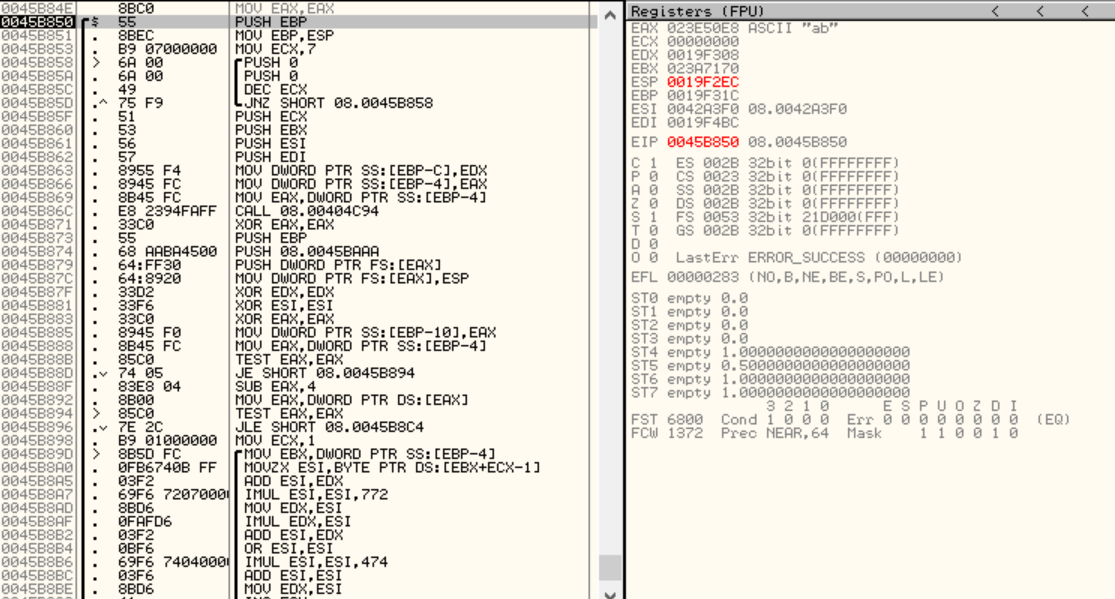
EAX 값과 EDX 값을 CMP 문으로 비교하는데 여기서 EAX는 내가 입력한 시리얼 값인 건 알겠는데 EDX의 값이 어디에서 나온 값인지 확인하기 위해 다시 디버깅했다.

다시 천천히 확인해보니 아래의 여러 개의 CALL MOV..문을 돌면서 내가 name으로 입력한 값이 어떠한 알고리즘에 의해 “5CBA- ..” 꼴로 바뀌어 EDX 값에 저장된다는 것 같다.





045BB9B의 CALL 함수 내부를 살펴보았다.



아래로 내려보면 뭔가 많은 연산이 이루어지는 루프문이 있었다.

여기서 이 다음부분은 한참 헤매다가 다른 분의 라업을 참고해봤는데 루프부분을 코드로 짜서 해결할 수 있다고 한다… 조금 더 공부한 다음에 다시 도전해봐야 할 것 같다.