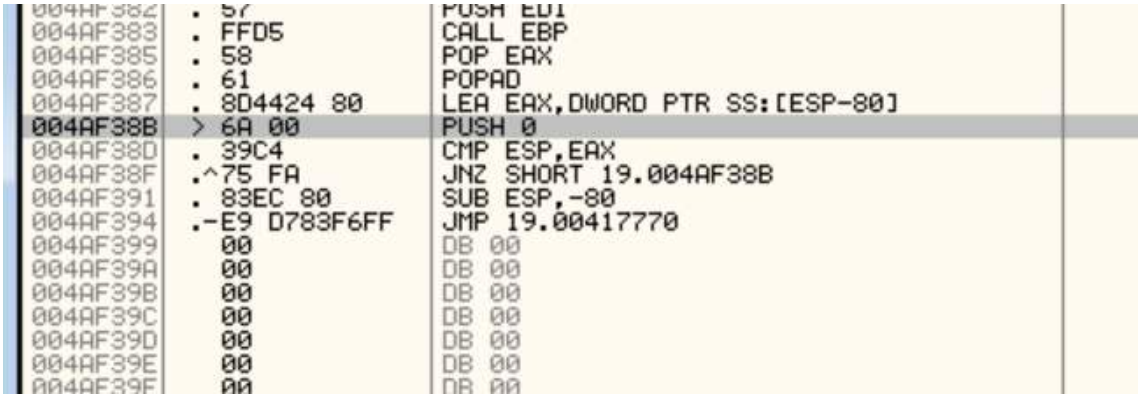
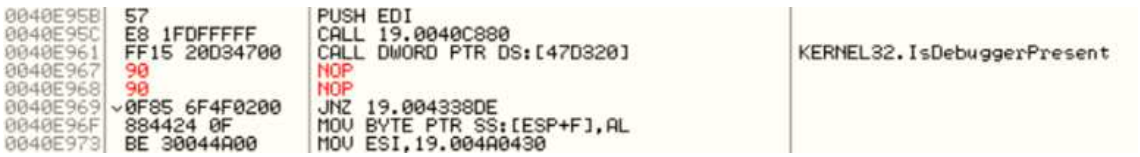
CodeEngn Basic RCE L19
: 이 프로그램은 몇 밀리세컨드 후에 종료 되는가

아마도 timeout과 관련된 문제인 것 같다.



올리디버거로 열어주었더니 오랜만에 PUSHAD가 보여서 command 찾기로 POPAD를 찾아주었고, 점프 관련 명령어가 2개가 있어서 살펴보니 JNZ는 004AF38B로 가면서 루프를 돈다. 즉 OEP로 가는 점프명령어는 JMP 00417770이다. 해당 주소에 BP를 걸어주고 F9 -> F8을 누르면 실제 코드로 이동한다.

실제코드에서 딱히 단서가 될만한 것이 없어서 search for -> all intermodular calls 눌러서 우클릭 -> sort by -> destination 으로 함수 별 탐색을 하였다.



방향성을 못잡고 있어서 검색을 해 본 결과 IsDebuggerPresent 함수가 있으면 해당 부분을 NOP으로 채워야 한다고 한다. 위 함수가 존재를 해서 NOP으로 채워주었다.

함수를 내리다 보면 timeGetTime을 찾을 수 있는데, 이 함수는 현재 시간을 불러오는 함수이다. 시간 관련 함수이므로 문제의 답을 구하는 것과 관련이 있을 것이라 생각했다.

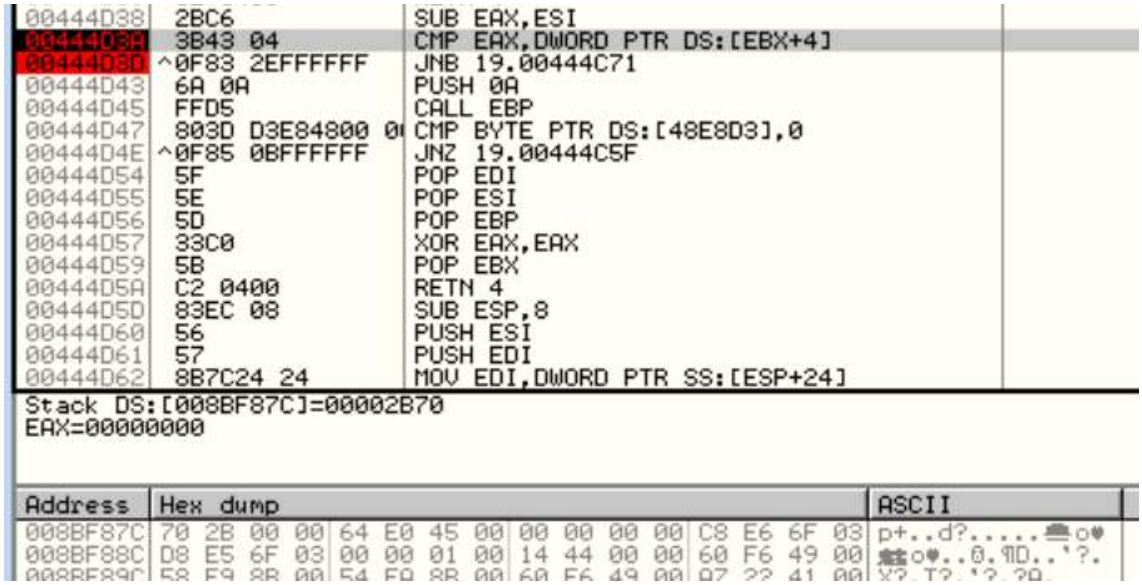
timeGetTime 부터 Sleep함수까지 살펴보면,

CALL EDI // EDI 호출 == timeGetTime함수 호출
MOV ESI, EAX // ESI = EAX : ESI에 EAX값 저장
JE 00444D54 // 0이면 00444D54로 JMP
.
.
CALL EDI // 다시 timeGetTime 함수 호출
CMP EAX, ESI // EAX - ESI
JNB 00444D38 // **Jump Not Below : 작지 않을 경우 jmp
=> EAX가 ESI보다 더 크면 JMP
SUB EAX, ESI // EAX = EAX - ESI
DEC EAX // EAX = EAX - 1
JMP 00444D3A // 해당 주소로 JMP

위에서 timeGetTime 함수를 2번 호출하고 있다. 즉 코드를 살펴보면 처음으로 호출한 부분에서 그때의 시각을 ESI에 저장하고, 2번 째 호출을 한 시각을 EAX에 저장을 해서 두 개를 비교를 하는데, 당연히 2번째 때 호출한 EAX가 ESI보다 클 수 밖에 없다. 그래서 JNB로 무조건 이동한다.

여기서 주목해야 하는 것은 JNB이다. EAX > ESI 면 해당 주소로 점프하라는건데 보통 타임아웃은 지정 시간이 현재시간보다 초과될 경우 발생한다. 그럼 EAX가 초과한 경우 해당 주소로 이동하라는 것 같아서 EAX가 경과된 시간이라고 추측했다. 그럼 00444D38주소에 가면 EAX가 어떻게 작동하는지 더 볼 수 있을 것이다.

SUB EAX, ESI // EAX = EAX - ESI
CMP EAX, DWORD PTR DS : [EBX+4] // EAX와 EBX+4주소의 데이터영역의 4바이트 값을 비교
정답으로 가는 KEY이다. 왜냐면 EAX - ESI는 2번째 호출 시각 - 1번째 호출 시각이기 때문이다. 아마 경과 시간이라고 생각된다. 근데 이 경과 시간을 비교하고 있으니 프로그램에서 지정해둔 타임아웃을 시간을 EBX+4주소의 4바이트가 가지고 있을 것이라고 추측된다.



BP를 걸고 실행시켜주면 해당 주소 DS 부분에 2B70이 쓰여져 있는 것을 볼 수 있다. (리틀엔디안이라서 HEX DUMP에서 오른쪽부터 읽어주어야 한다.) 2B70을 10진수로 바꾸면 11120이고 이 프로그램은 11120 밀리세컨드 후에 종료된다는 것을 알 수 있다.

ms = 10의 -3승 s이므로 11.12초 이후에 종료된다.