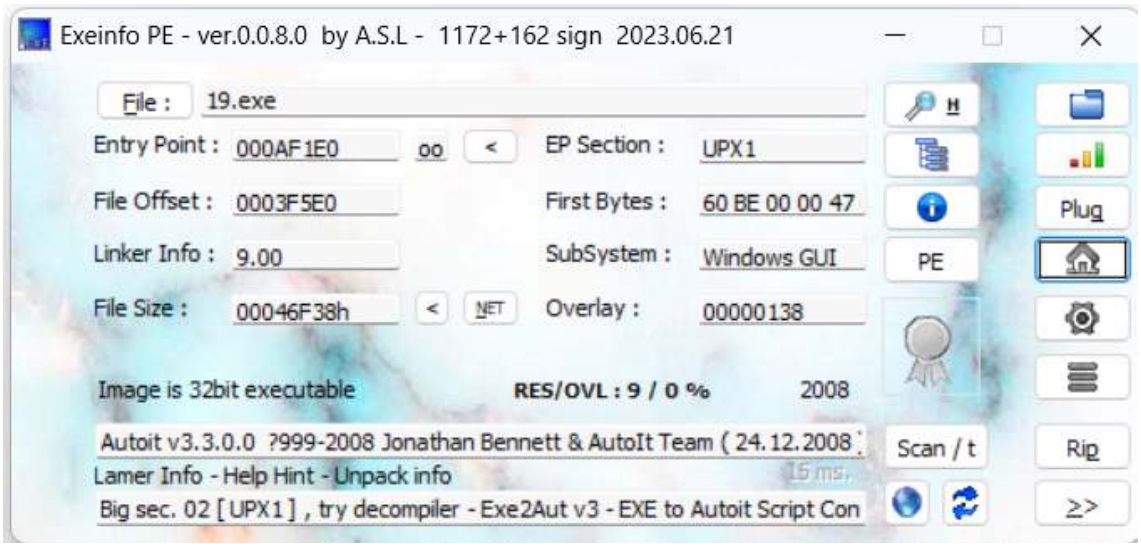
CodeEngn Basic RCE

강 보 성

[19]

풀이 : 19.exe가 upx로 패킹되어있기 때문에 원활한 분석을 위해 언패킹한다.

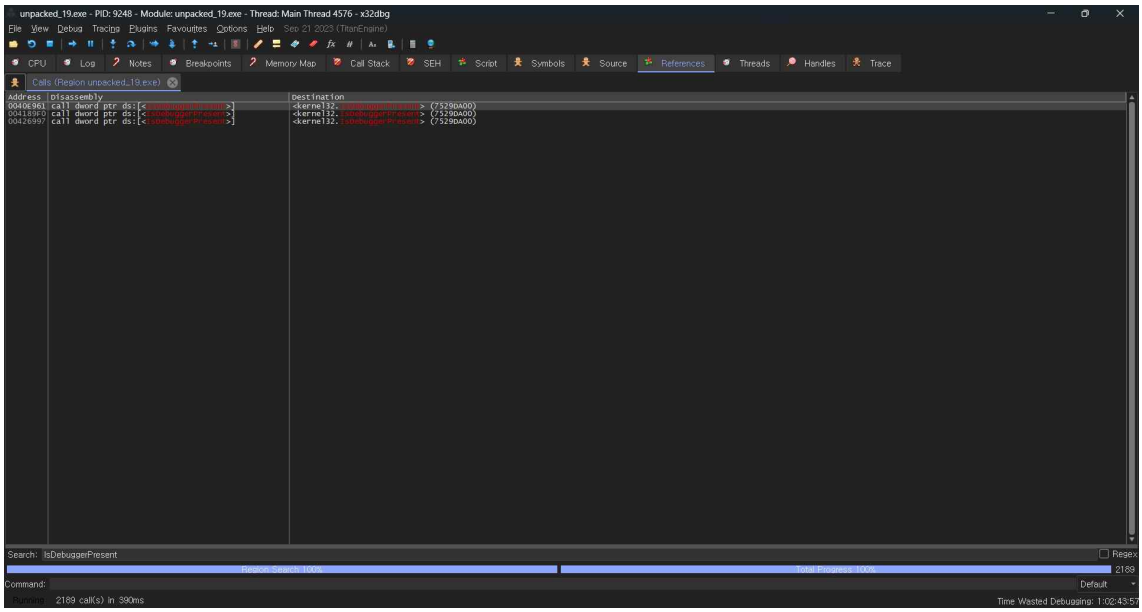


<Exeinfo PE에서 확인한 19.exe 정보>

언패킹 후 디버깅을 하면 아래와 같은 메시지를 확인할 수 있다. 이 메시지를 통해 19.exe에 안티 디버깅 기법이 적용되어 있음을 알 수 있다.

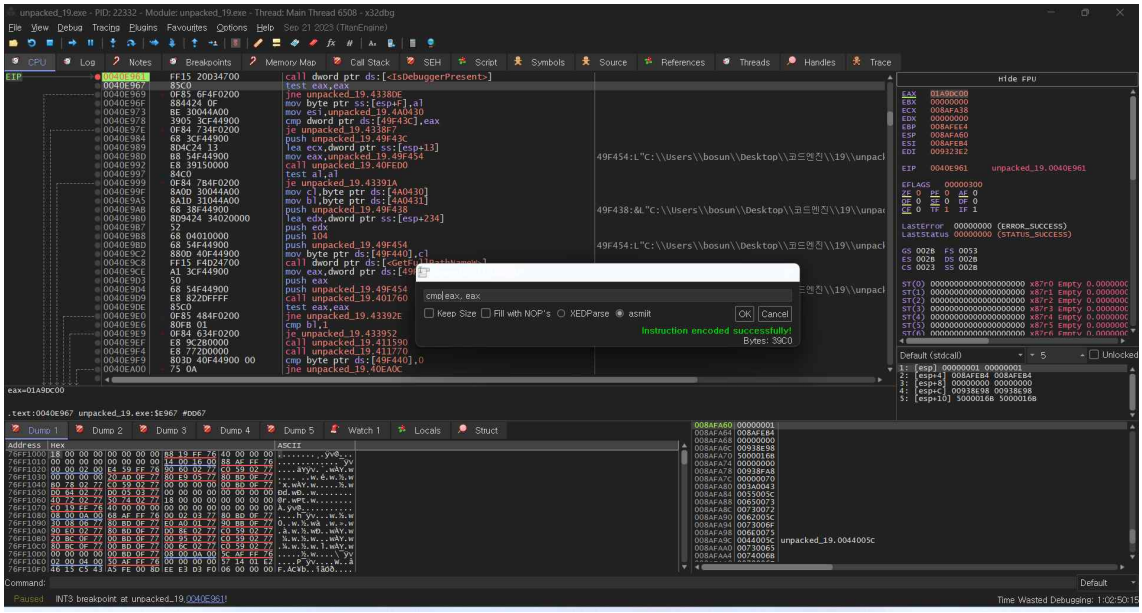
<디버깅 감지 메시지>

원활한 디버깅을 위해 디버깅을 감지하는 함수를 찾아서 무력화시켜야 한다. 함수 검색으로 IsDebuggerPresent 함수가 존재함을 알 수 있다.



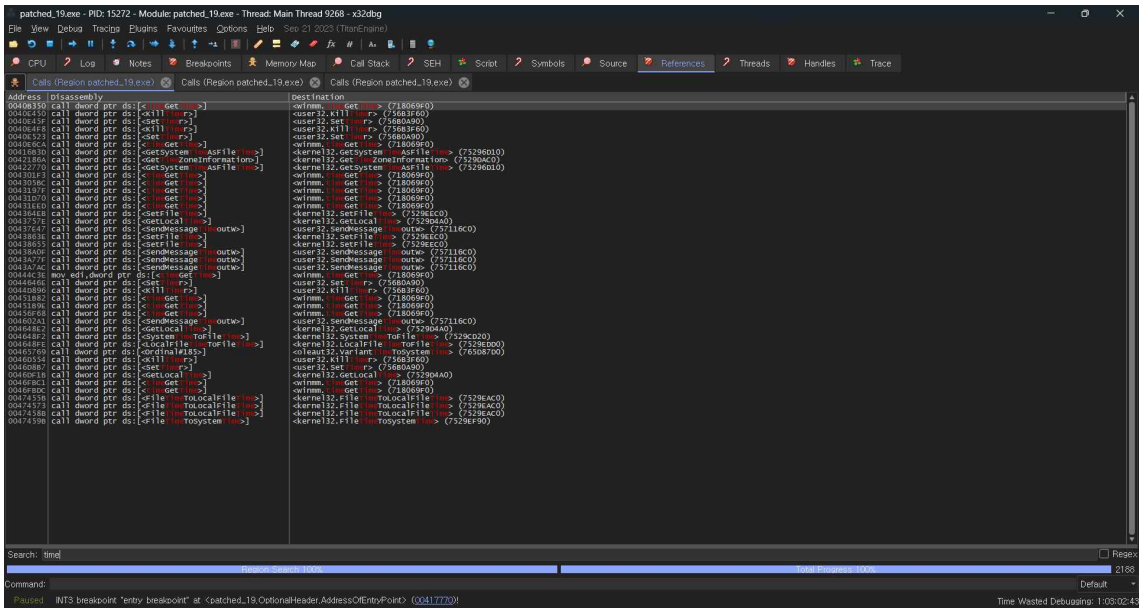
<IsDebuggerPresent 함수 검색>

해당 함수들에 브레이크 포인트를 걸고 다시 실행한다. 그러면 IsDebuggerPresent 함수 호출 이후 jne 분기문을 발견할 수 있다. IsDebuggerPresent 함수를 무력화시키기 위해 test eax, eax를 cmp eax, eax로 패치한다.



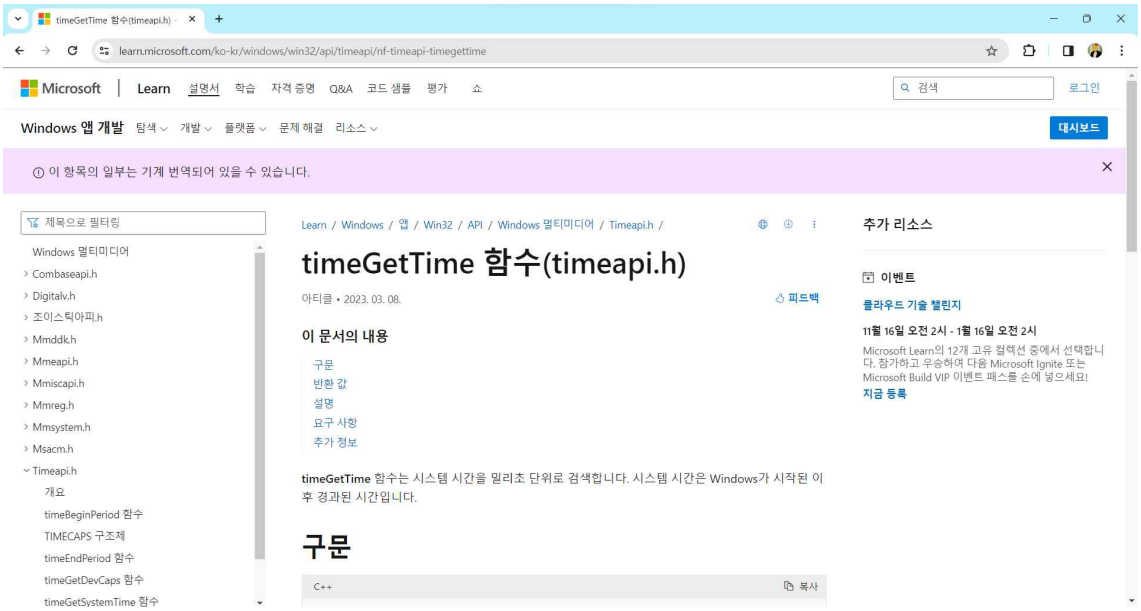
<cmp eax, eax 패치>

본격적으로 문제를 해결하기 위해 시간에 대한 함수를 검색한다. 많은 함수들이 보이는데, 그 중에서 timeGetTime 함수가 눈에 띈다.



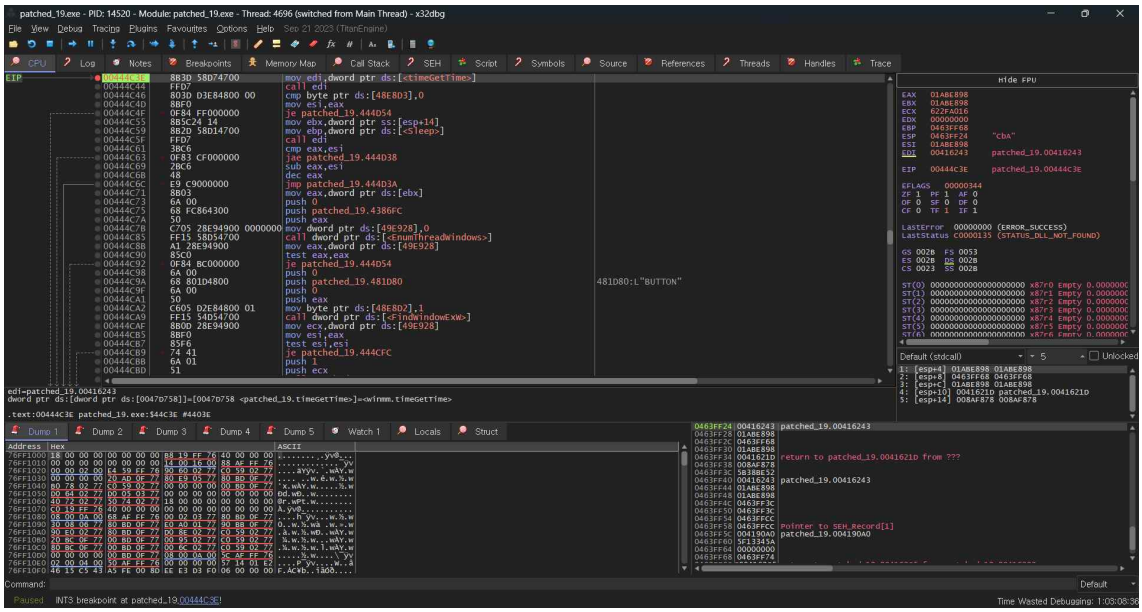
<이름에 'time'을 포함하는 함수 검색>

검색을 통해 timeGetTime 함수에 대해 알아보자. 이 함수가 문제 해결에 중요한 역할을 할 것으로 추측된다.



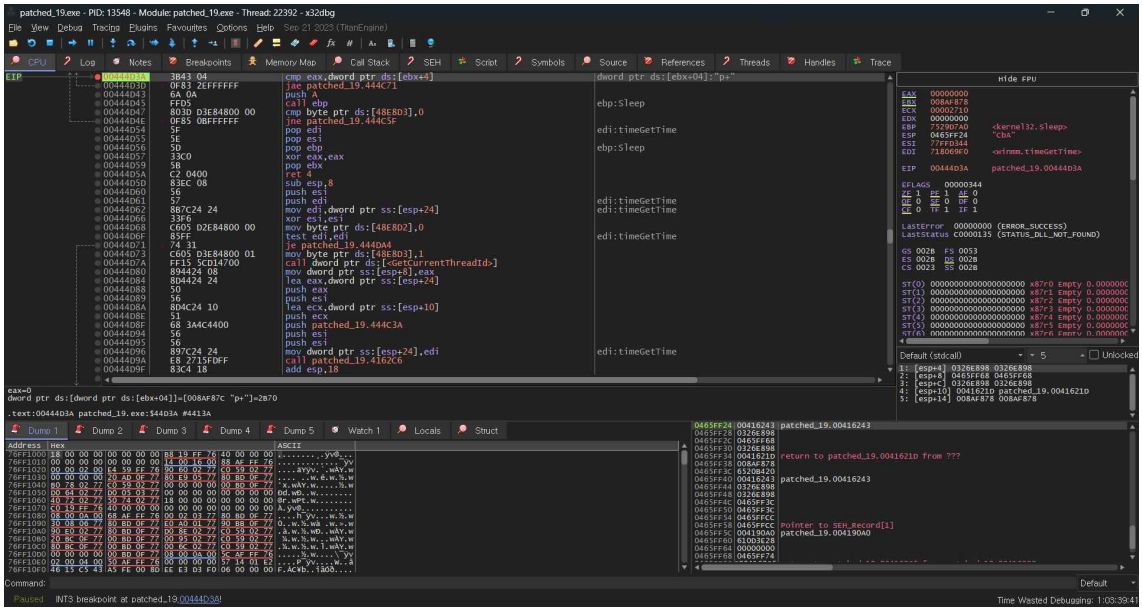
<timeGetTime 함수 검색>

따라서 해당 함수를 포함하는 모든 코드에 브레이크 포인트를 걸어두고 다시 실행한다. 그러면 아래와 같은 화면을 볼 수 있다.



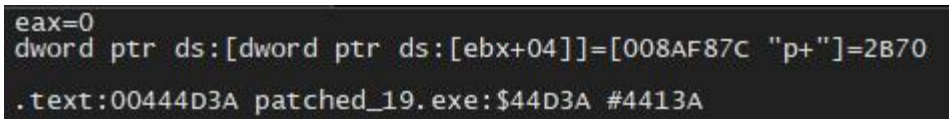
<timeGetTime 함수 호출>

코드를 분석하자면, 먼저 edi에 timeGetTime 함수를 저장하여 한번 호출한다. 이렇게 해서 구한 시간을 esi에 저장하며 이것은 프로그램을 처음 실행한 시간이 된다. 이후 edi를 다시 호출하여 시간을 구한다. 이때 시간은 eax에 저장된다. sub eax, esi를 통해 마지막으로 구한 시간에서 처음 구한 시간을 뺄셈하여, 프로그램이 처음 실행되고 경과된 시간을 eax에 저장한다. 밑으로 내리며 코드를 더 분석하다보면 cmp eax, dword ptr ds:[ebx+4] 구문과 jae 분기문이 보인다.



<cmp 구문과 jae 분기문>

eax 값과 ds:[ebx+4]에 저장된 값을 비교하여 같다면 jae 분기문을 따라 점프를 하고, 그렇지 않다면 앞 과정으로 돌아가서 경과 시간을 다시 구한다. 결국 문제에서 요구하는 시간은 ds:[ebx+4]에 저장되어있을 것으로 추정됨으로 해당 값을 찾는다.



<ds:[ebx+4]에 저장된 0x2B70>

0x2B70을 10진수로 변환하면 11120임을 알 수 있다.

<16진수 2B70을 10진수로 변환>

답 : 11120