악성코드 분석 보고서

(sand-reversingwithlena-tutorials)

2025.08.06

19

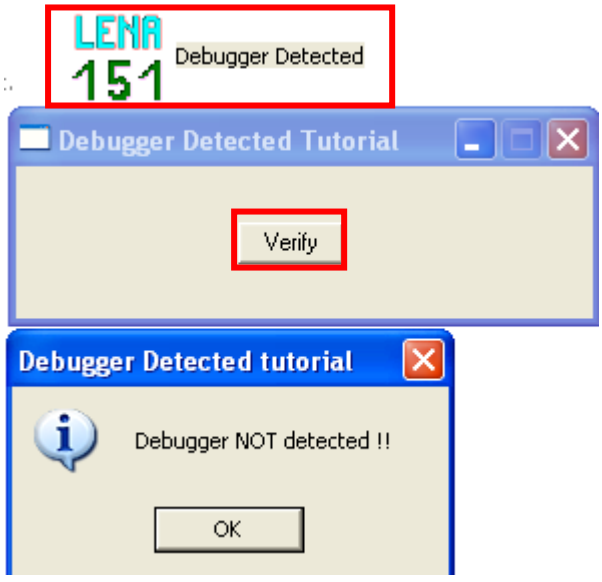
1. 문제

1. ReverseMe.A

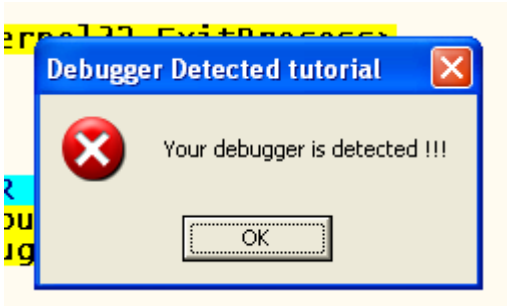
클릭 시 성공했다는 창이 나온다.

올리디버그 실행 시 키 파일이 유효하지 않다고 나온다.

2. Debugger Detected



'Debugger Detected'를 누르면 밑에 창이 나오고 'Verify'를 누르면 'Debuuger NOT detected' 창이 나온다.

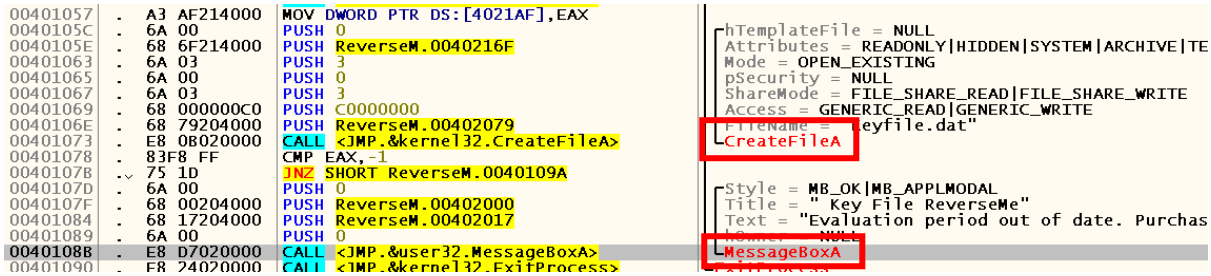


하지만 올리디버그로 실행 시 구문이 다른 창이 나오는 걸 알 수 있다.

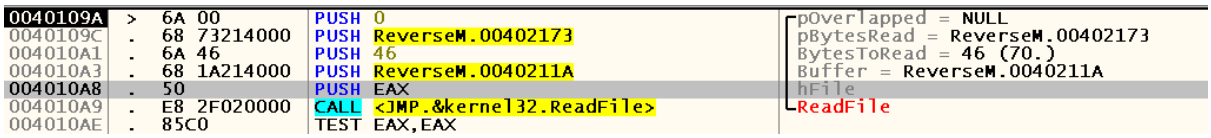
왜 서로 창이 다르게 나오는 지 알아봐야한다.

2. 해결 방법

1. ReverseMe.A



Keyfile.dat라는 파일을 여는 함수를 볼 수 있는데 지금 Keyfile.dat이 있기 때문에 레지스터 키 값을 구매해달라는 메시지 박스는 넘어가는 걸 알 수 있다.

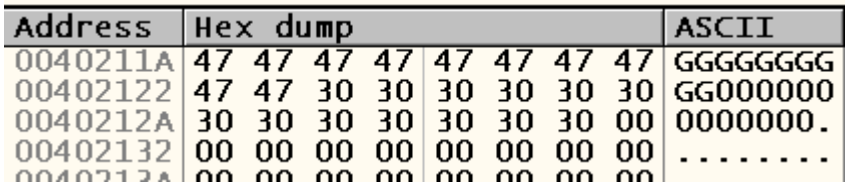


ReadFile함수가 실행되는데 keyfile.dat의 값들이 0x0040211A에 저장되는 걸 볼 수 있다.

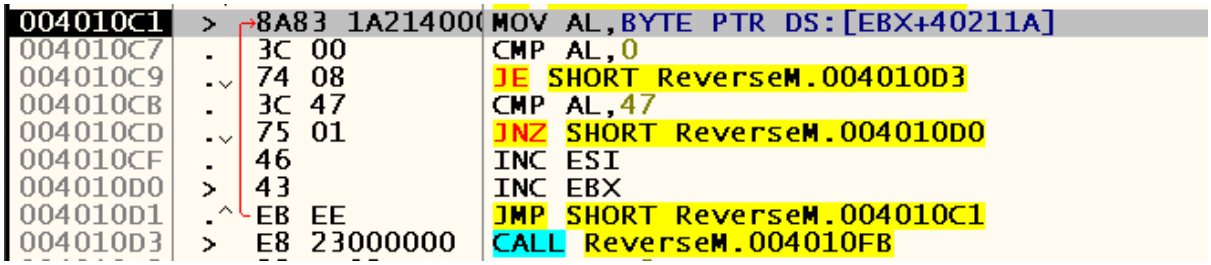
* pBytesRead와 Buffer의 차이

pBytesRead : 함수 호출 후 실제 읽혀 진 데이터의 크기를 저장할 메모리의 주소

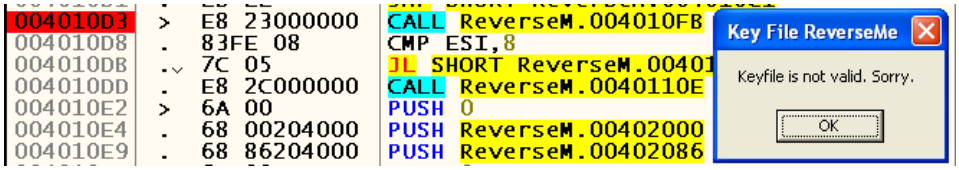
Buffer : 읽어온 내용을 저장할 메모리의 시작 주소를 설정한다.



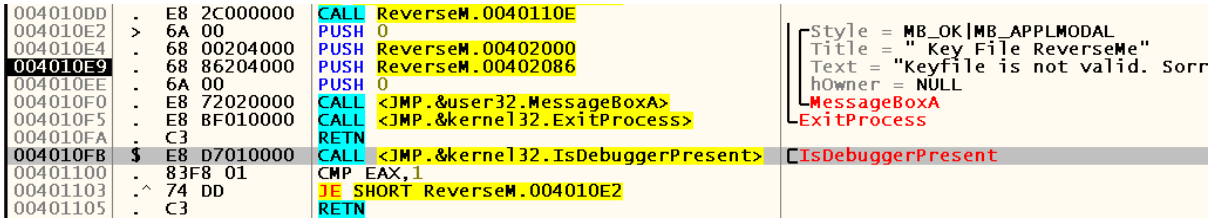
해당 덤프 창에 들어가서 보면 값이 저장된 걸 볼 수 있다.



이 부분은 keyfile.dat에서 'G'문자가 몇 개인지 세는 과정이다.



이 부분에서 창이 나오는 걸 볼 수 있고 'F7'을 눌러 함수에 들어가본다.

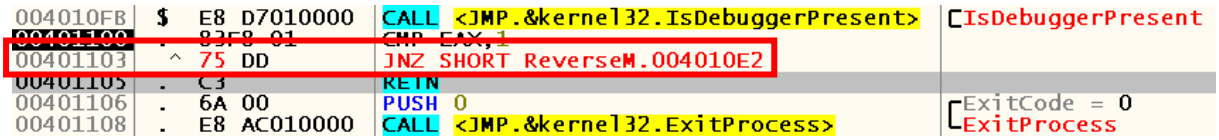


IsDebuggerPresent함수로 넘어간 후 MessageBoxA로 넘어가는 걸 알 수 있다. 메시지 박스로 넘어가기 전에 EAX를 1과 비교하는 걸 볼 수 있는데 IsDebuggerPresent함수가 **호출 프로세스가 사용자 모드 디버거에 의해 디버그되고 있는지 여부를 확인**하는 함수라고 나와있고

반환 값으로는 현재 프로세스가 디버거의 컨텍스트에서 실행 중인 경우 반환 값은 0이 아닌 값, 실행되고 있지 않으면 반환 값은 0이다.

하지만 올리디버그에서 실행되고 있으므로 반환 값은 0이 아닌 값이고 레지스터를 보면 EAX = 1로 나오는 걸 볼 수 있다.

그럼 성공했다는 창이 나오기 위해서는 EAX = 0으로 되거나 JE -> JNE로 바껴야 성공 창이 나오는 걸 알 수 있다.

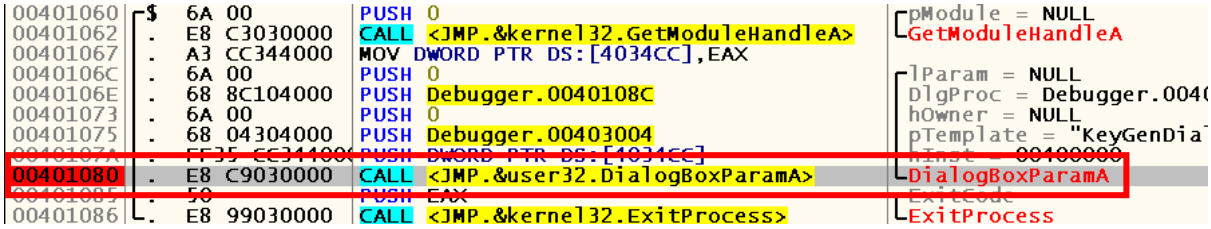


JE 004010E2 -> JNE 004010E2로 바꾸고 저장하고 올리디버그로 실행해보면

해당 창이 나오는 걸 볼 수 있다.

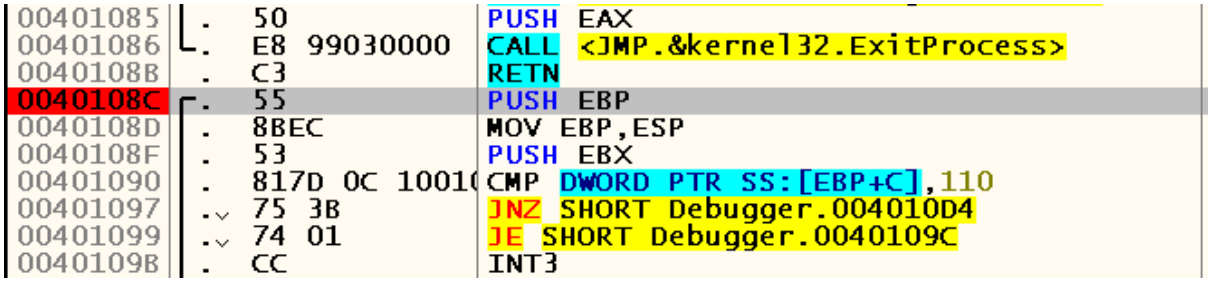
문제 해결!

2. Debugger Detected

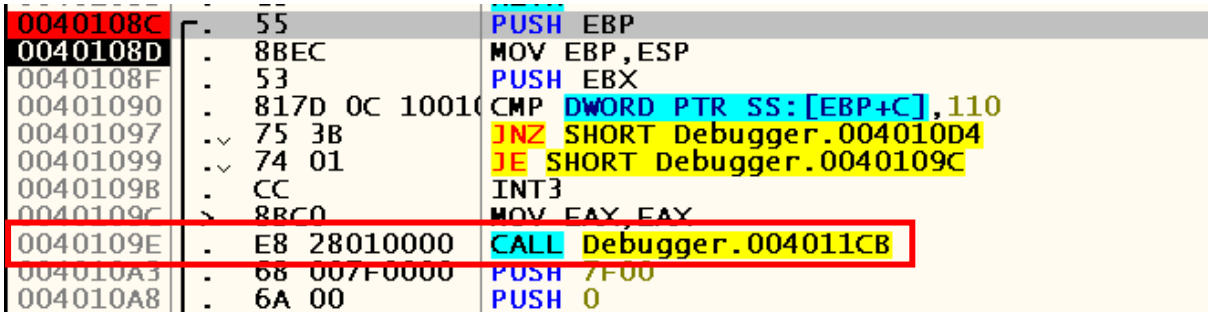


0x00401080에서 실행 시 창이 나오는 걸 알 수 있다.

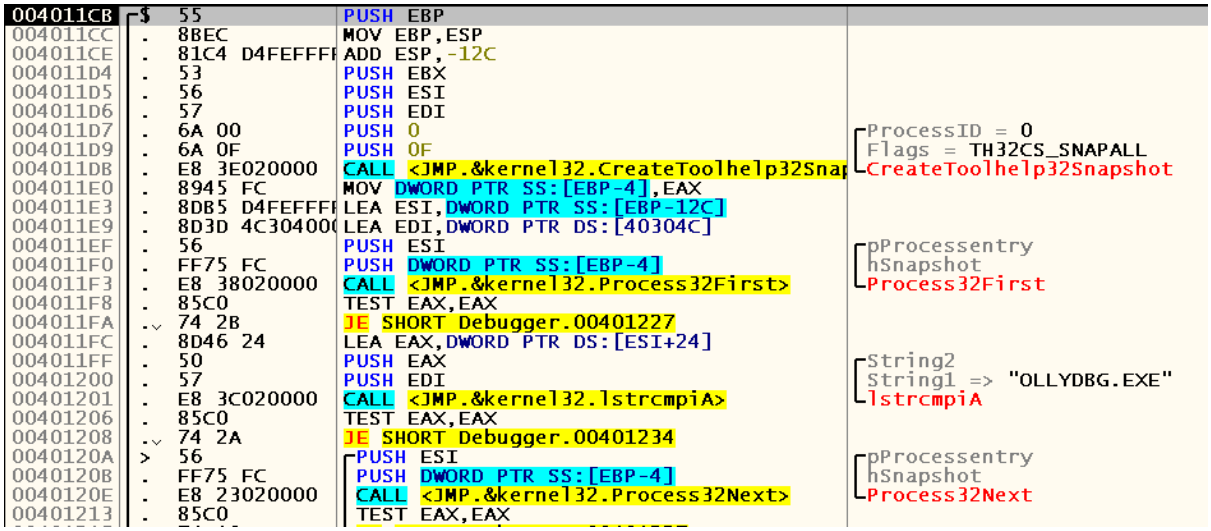
DialogBoxParamA 를 msdn 에서 찾아보면 DLGPROC 인자가 **대화 상자 프로시저에 대한 포인터**라고 나온다. 즉, 0x0040108C 에 있는 함수가 실행된다는 뜻이다.



0x0040108C 에 BP 를 걸고 'ctrl+F2'로 재실행 후 해당 주소로 넘어오고 'F8'로 실행해보면 0x00401097 에서 점프하는 걸 알 수 있다.



계속 실행 해보면 다시 0x0040108C 여기에 도달하는 걸 알 수 있고 004011CB 에 도달했을 때 해당 주소로 넘어가야하는데 다시 0x0040108C 여기에 가는 걸 알 수 있다. 즉, 004011CB 를 부르는 함수 쪽이 이상하다는 걸 알 수 있고 이 함수를 분석해야한다.



'F7' 이용해서 들어가보면 여러가지 msdn 함수가 나온다.

* msdn 함수 설명

CreateToolhelp32snapshot : 지정된 프로세스의 스냅샷 이러한 프로세스에서 사용되는 힙, 모듈 및 스레드를 사용

반환 값 -> 함수가 성공하면 지정된 스냅샷 열린 핸들을 반환

Process32First : 시스템 스냅샷 발생한 첫 번째 프로세스에 대한 정보를 검색

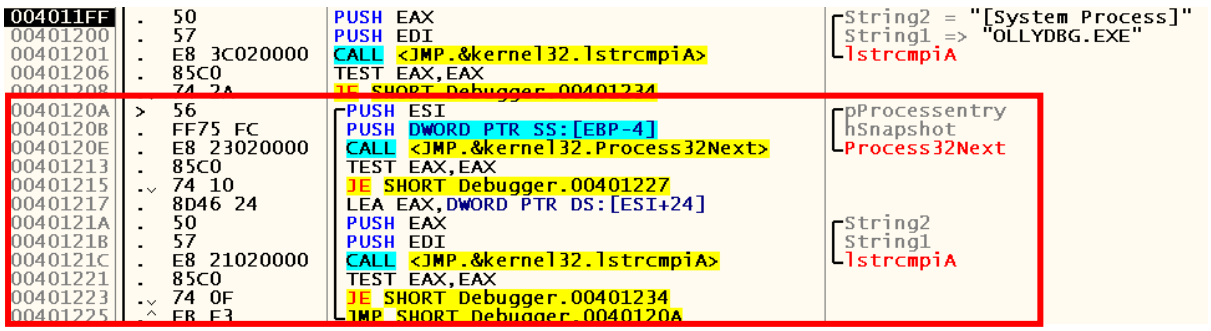
반환 값 -> 프로세스 목록의 첫 번째 항목이 버퍼에 복사되었으면 **TRUE(1)** 를 반환

lstrcmpiA : 두 문자 문자열을 비교

반환 값 -> lpString1 가리키는 문자열이 lpString2 가리키는 문자열보다 작으면 반환 값은 음수 아니면 양수

Process32Next : 시스템 스냅샷 기록된 다음 프로세스에 대한 정보를 검색

반환 값 -> 프로세스 목록의 첫 번째 항목이 버퍼에 복사되었으면 **TRUE(1)** 를 반환

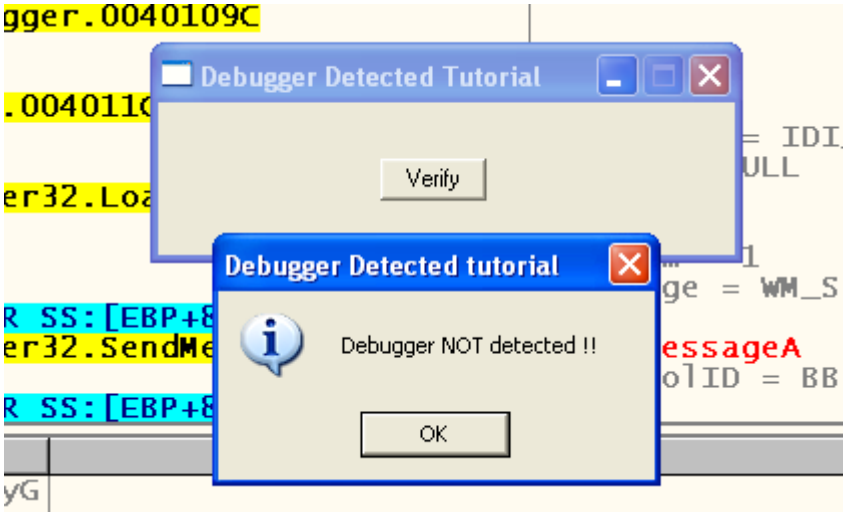


lstrcmpiA 에서 프로세스 문자열을 비교하는 걸 볼 수 있는데 'OLLYDBG.EXE' 올리디버그를 찾는 걸 알 수 있다. (그래서 올리디버그에서 실행하면 다른 창이 나옴)

lstrcmpiA 함수는 EAX = 1 을 반환하고 루프를 도는 걸 알 수 있다. 아마 현재 실행 중인 프로세스 중 OLLYDBG.EXE 가 나올 때까지 해당 루프를 도는 거 같다.

```
00401208  .v 74 2A        JE SHORT Debugger.00401234
0040120A  >  56           PUSH ESI                          pProcessentry
0040120B  .  FF75 FC      PUSH DWORD PTR SS:[EBP-4]         hSnapshot
0040120E  .  E8 23020000  CALL <JMP.&kernel32.Process32Next> Process32Next
00401213  .  85C0         TEST EAX,EAX
00401215  .v 74 10        JE SHORT Debugger.00401227
00401217  .  8D46 24      LEA EAX,DWORD PTR DS:[ESI+24]
0040121A  .  50           PUSH EAX                          String2
0040121B  .  57           PUSH EDI                          String1
0040121C  .  E8 21020000  CALL <JMP.&kernel32.lstrcmpiA>    lstrcmpiA
00401221  .  85C0         TEST EAX,EAX
00401223  .v 74 0F        JE SHORT Debugger.00401234
00401225  .^ EB E3        JMP SHORT Debugger.0040120A
00401227  >  FF75 FC      PUSH DWORD PTR SS:[EBP-4]         hObject
0040122A  .  E8 E9010000  CALL <JMP.&kernel32.CloseHandle>  CloseHandle
0040122F  .  5F           POP EDI
00401230  .  5E           POP ESI
00401231  .  5B           POP EBX
00401232  .  C9           LEAVE
00401233  .  C3           RETN
00401234  >  6A 10        PUSH 10                           Style = MB_OK|MB_ICONHAND|MB_APPLMODAL
00401236  .  68 11304000  PUSH Debugger.00403011            Title = "Debugger Detected tutorial
0040123B  .  68 58304000  PUSH Debugger.00403058            Text = "Your debugger is detected !!!"
00401240  .  FF75 08      PUSH DWORD PTR SS:[EBP+8]         hOwner
00401243  .  E8 24020000  CALL <JMP.&user32.MessageBoxA>    MessageBoxA
```

OLLYDBG.EXE 를 찾게 되면 0x00401223 에서 점프하는 걸 알 수 있고 그럼 디버거 사용하고 있다는 창이 나오게 되니까 점프를 하면 안된다.



안넘어가게 하기 위해서는 JE 00401234 -> NOP 으로 바꾸고 저장해서 올리디버그로 실행해보면 해당 창이 나오는 걸 알 수 있다.

문제 해결!