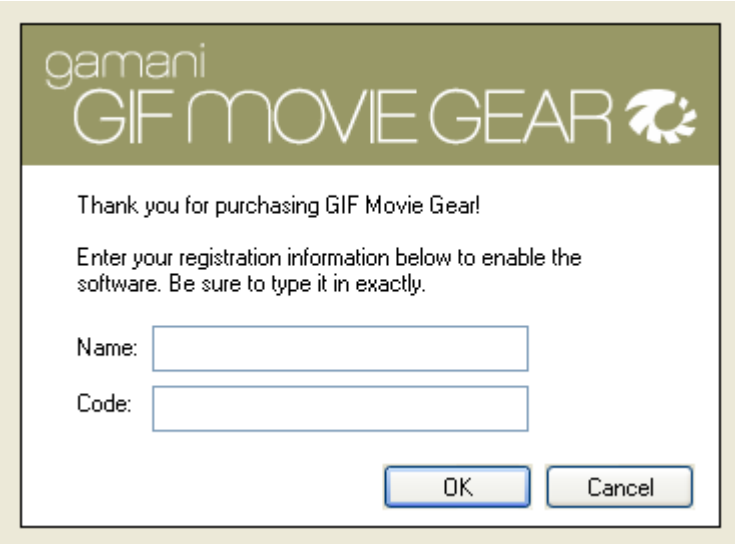
악성코드 분석 보고서

(sand-reversingwithlena-tutorials)

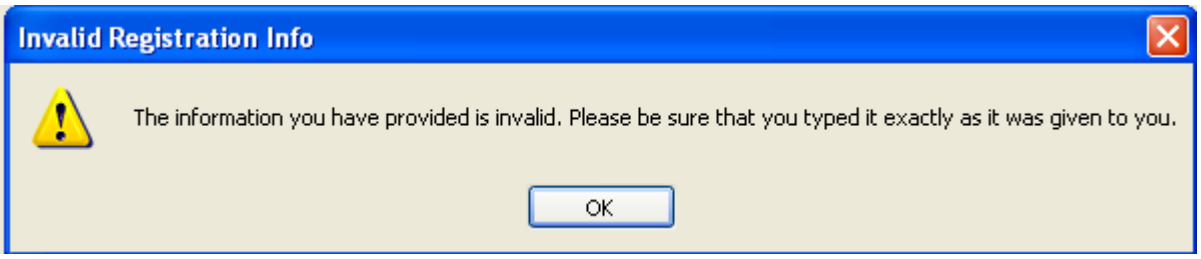
2025.07.21

16

1. 문제



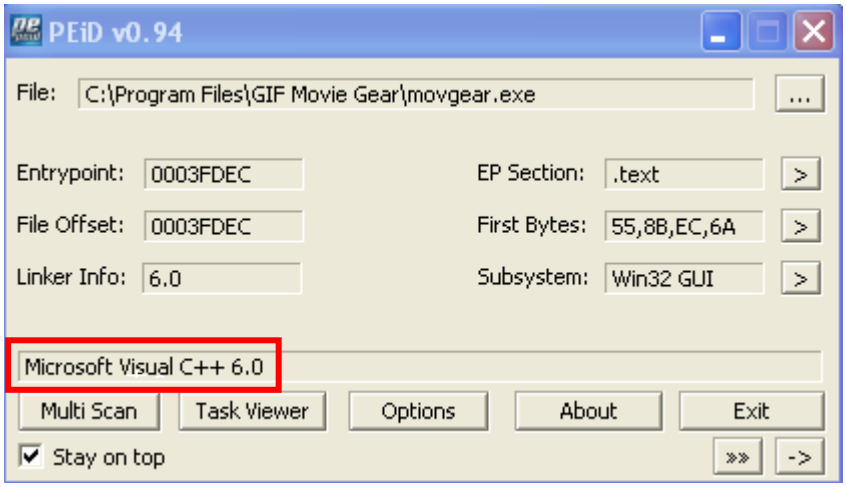
레지스트리 키 등록



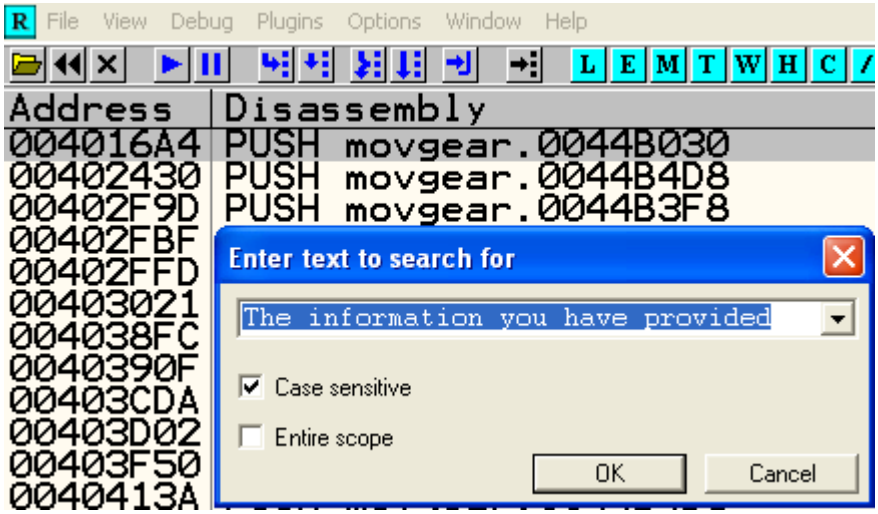
키 입력 틀렸을 때 나오는 구문

프로그램 나갈 때 나오는 잔소리 삭제하기 -> 아마 키를 등록하면 없어질거로 예상

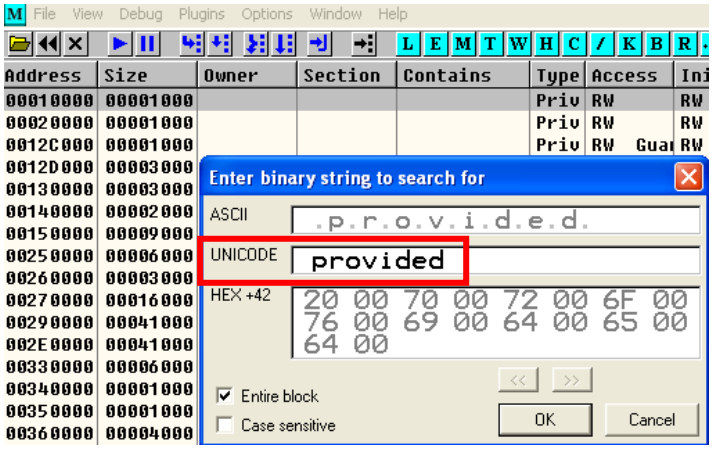
2. 해결 방법

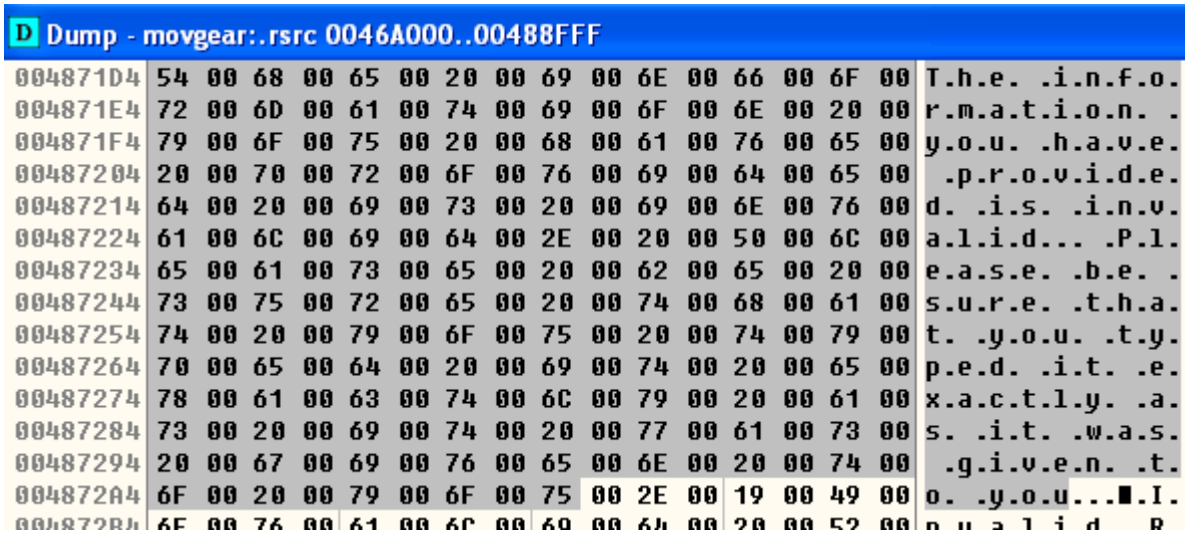


PEID로 확인 시 C++로 만든 파일이라는 것을 알 수 있다.
키가 틀린 경우 'The information you have provided is invalid. Please be sure that you typed it exactly as it was given to you.'라고 뜨는데 이걸 이용해서 찾아본다.

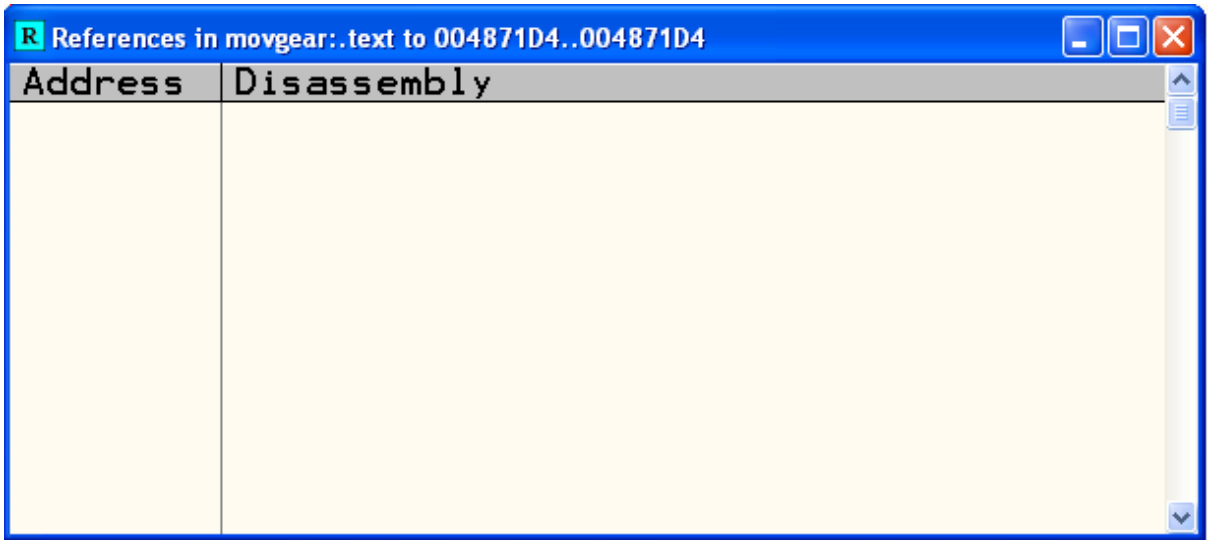


Search for > All referenced text strings 눌러서 해당 구문을 찾아줬는데 찾을 수 없다.



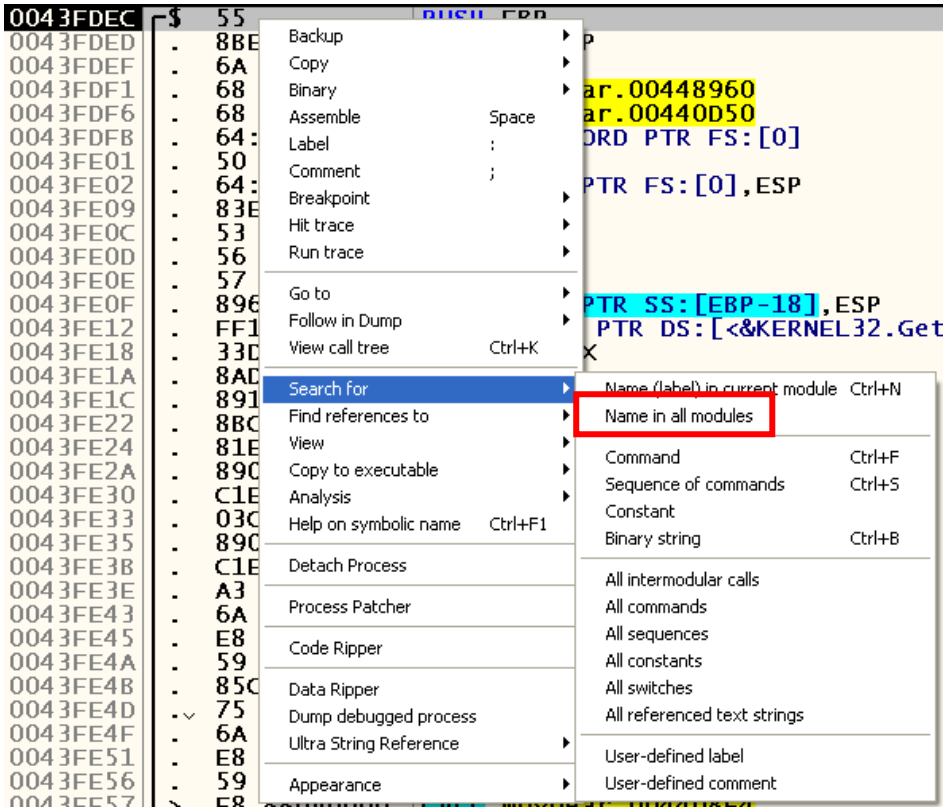


Memory map에 들어가 UNICODE에 넣어보면 나오는 것을 볼 수 있다.

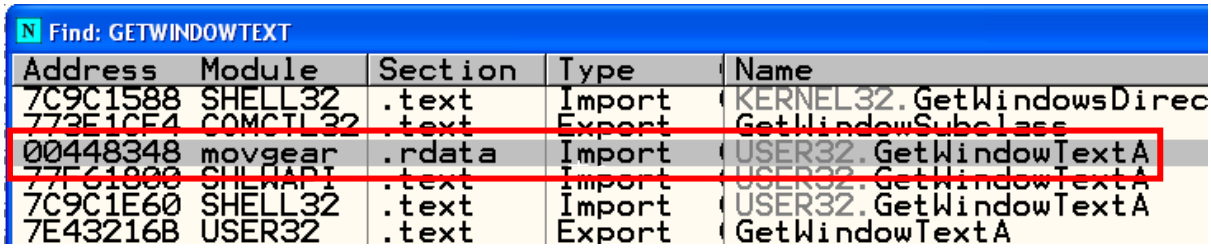


0x004871D4 메모리에 들어가서 참조하는 것을 찾아보면 아무것도 없는 걸 볼 수 있다. 그럼 이 방법이 아닌 다른 방법을 찾아야한다.

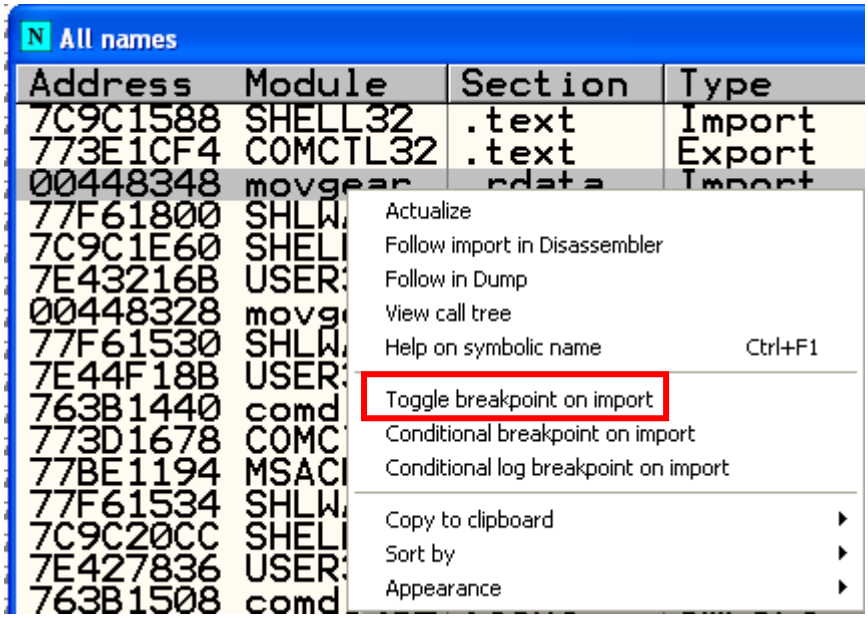
키를 입력하는 곳은 GetwindowTextA 함수를 사용했을 거라고 추정된다.



Search for > Name in modules에 들어가서 GetwindowTextA 를 찾아준다.



총 4개가 있는데 그 중 모듈이 movgear에 브레이크를 걸어준다.



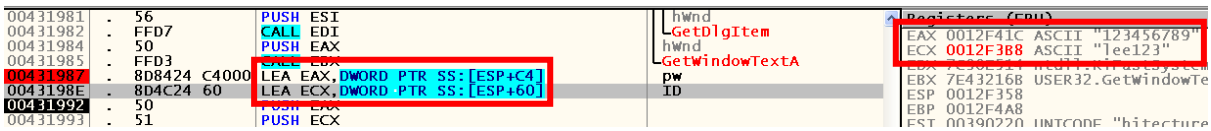
※ toggle breakpoint on import & conditional breakpoint on import 차이

toggle breakpoint on import: 조건 없이 무조건 걸림

conditional breakpoint on import: 조건식을 만족하는 경우에만 멈춤.

이렇게 브레이크를 걸어주고 'F9'으로 실행해준다.

여기에 아무 값이나 넣고 실행해준다.



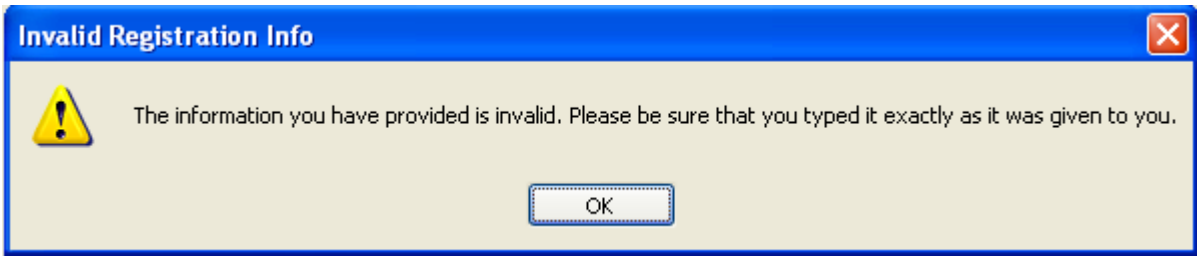
실행하다보면 레지스터에 code랑 name을 EAX와 ECX에 옮겨주는 코드를 볼 수 있다. 이 부분은 브레이크를 걸어주고 실행해준다.

```
00431993   .   51              PUSH ECX
00431994   .   E8 F7FBFFFF    CALL movgear.00431590
00431999   .   83C4 08         ADD ESP,8
0043199C   .   85C0            TEST EAX,EAX
0043199E   .v  0F84 AD000000   JE movgear.00431A51
004319A4   .   8D5424 10       LEA EDX,DWORD PTR SS:[ESP+10]
004319A8   .   8D4424 0C       LEA EAX,DWORD PTR SS:[ESP+C]
004319AC   .   52              PUSH EDX
```

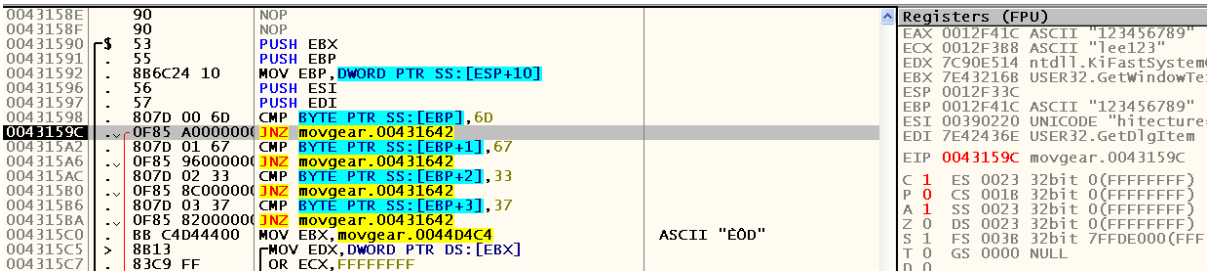
```
00431A4E   .   C2 1000         RETN 10
00431A51   >   6A 30           PUSH 30
00431A53   .   68 159D0000     PUSH 9D15
00431A58   .   68 149D0000     PUSH 9D14
00431A5D   .   56              PUSH ESI
00431A5E   .   E8 2DD4FDFF     CALL movgear.0040EE90
00431A63   .   83C4 10         ADD ESP,10
00431A66   .   68 4F040000     PUSH 44F
00431A6B   .   56              PUSH ESI
00431A6C   .   FFD7            CALL EDI
00431A6E   .   50              PUSH EAX
00431A6F   .   FF15 B4834400   CALL DWORD PTR DS:[<&USER32.SetFocus>]
00431A75   >   5F              POP EDI
00431A76       5E              POP ESI
```

```
hWnd
SetFocus
Default case of switch
```

실행하다보면 0x0043199E에서 EAX = 0으로 바뀌고 JE분기 때문에 넘어가고 0x00431A5E에서 함수가 실행되어 밑에 있는 창이 실행 되는 것을 볼 수 있다.



그럼 JE분기가 실행하기 전에 0x00431994함수를 브레이크 걸어서 다시 실행해준다.

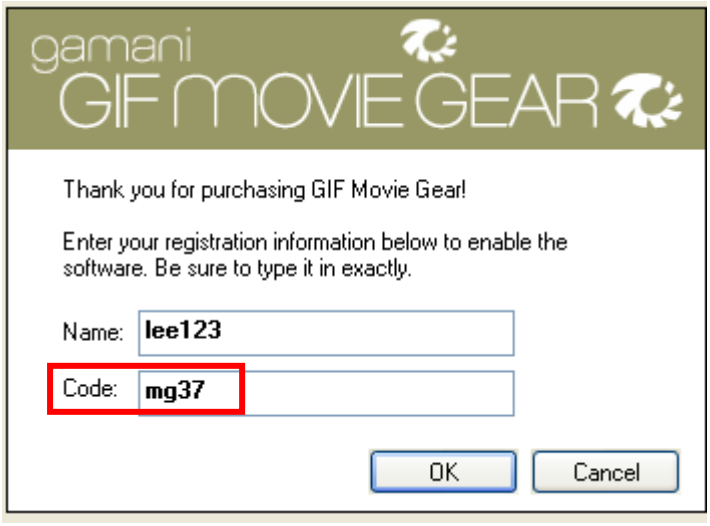



```
00431641   .   C3              RETN
00431642   >   5F              POP EDI
00431643   .   5E              POP ESI
00431644   .   5D              POP EBP
00431645   .   33C0            XOR EAX,EAX
00431647   .   5B              POP EBX
00431648   .   C3              RETN
```

함수를 들어가서 보면 0x0043159C함수에서 이 쪽 부분으로 넘어오는 것을 볼 수 있고 마찬가지로 모든 JNZ분기가 여기로 넘어오는 것을 볼 수 있다. 근데 EBP가 입력한 Code를 CMP로 비교하는 것을 볼 수 있는데 아마 비교하는거와 Code가 같아야 이 분기문을 넘어가는 것을 볼 수 있다.

```
0043160A  .  84D2          TEST DL,DL
0043160C  .v 74 26         JE SHORT movgear.00431634
0043160E  >  0FBED2        MOVSX EDX,DL
00431611  .  41            INC ECX
00431612  .  0FAFD1        IMUL EDX,ECX
00431615  .  03F2          ADD ESI,EDX
00431617  .  81FE BE170000 CMP ESI,17BE
0043161D  .v 7E 06         JLE SHORT movgear.00431625
0043161F  .  81EE BE170000 SUB ESI,17BE
00431625  >  83F9 0A       CMP ECX,0A
00431628  .v 7E 02         JLE SHORT movgear.0043162C
0043162A  .  33C9          XOR ECX,ECX
0043162C  >  8A57 01       MOV DL,BYTE PTR DS:[EDI+1]
0043162F  .  47            INC EDI
00431630  .  84D2          TEST DL,DL
00431632  .^ 75 DA         JNZ SHORT movgear.0043160E
00431634  >  3BF0          CMP ESI,EAX
00431636  .v 75 0A         JNZ SHORT movgear.00431642
```

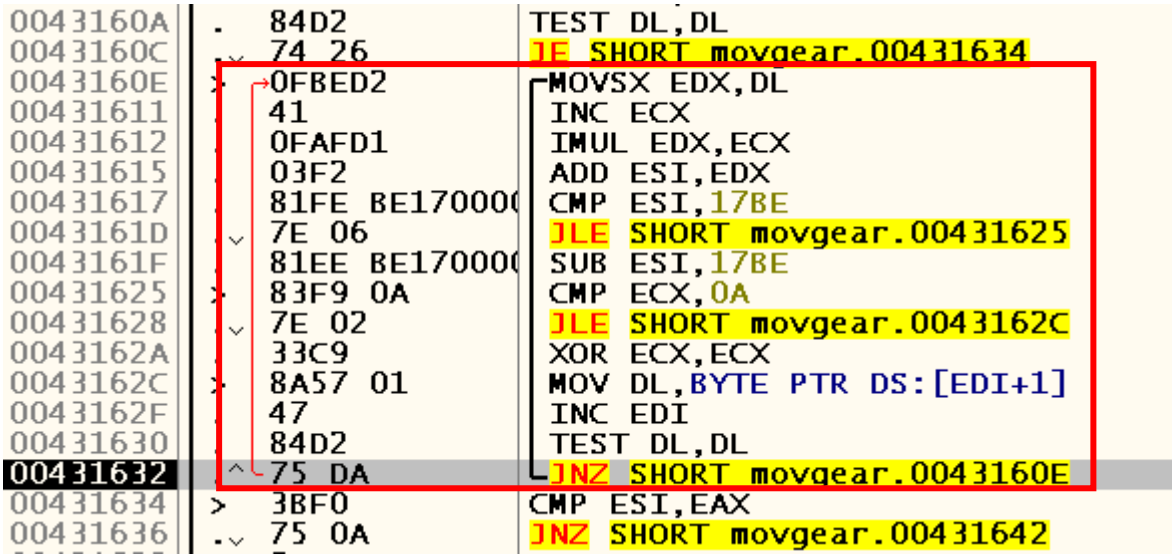
옆에 보면 숫자가 있는데 아스키 코드표를 이용해서 보면 60 = m, 67 = g, 33 = 3, 37 = 7로 비밀번호는 mg37로 추정할 수 있다.



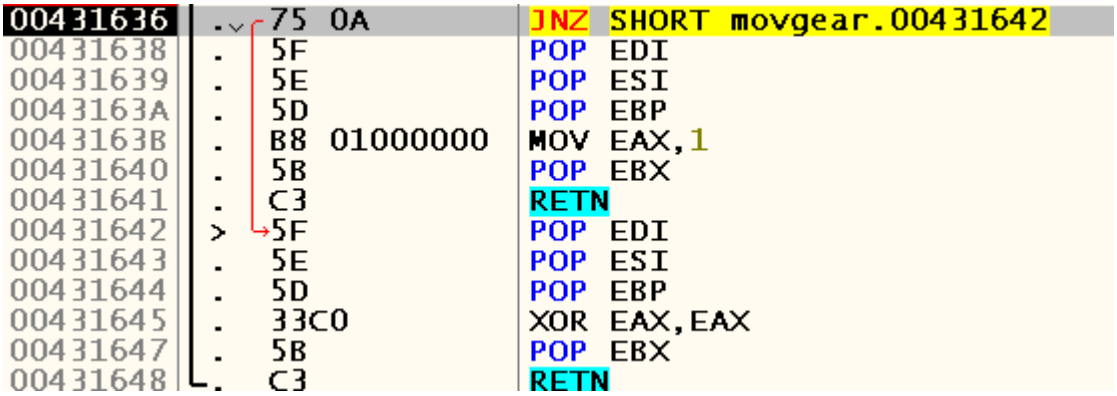
이걸 사용해서 다시 실행해본다.

```
00431598  .  807D 00 6D    CMP BYTE PTR SS:[EBP],6D
0043159C  .v 0F85 A0000000 JNZ movgear.00431642
004315A2  .  807D 01 67    CMP BYTE PTR SS:[EBP+1],67
004315A6  .v 0F85 96000000 JNZ movgear.00431642
004315AC  .  807D 02 33    CMP BYTE PTR SS:[EBP+2],33
004315B0  .v 0F85 8C000000 JNZ movgear.00431642
004315B6  .  807D 03 37    CMP BYTE PTR SS:[EBP+3],37
004315BA  .v 0F85 82000000 JNZ movgear.00431642
004315C0  .  BB C4D44400   MOV EBX,movgear.0044D4C4
004315C5  >  8B13          MOV EDX,DWORD PTR DS:[EBX]
004315C7  .  83C9 FF       OR ECX,FFFFFFFF
004315CA  .  8BFA          MOV EDI,EDX
```

그럼 전부 넘어오는 것을 볼 수 있다. 다시 실행하다보면

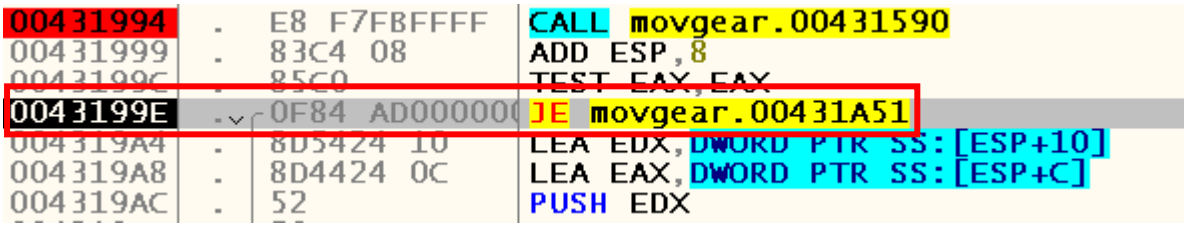


레지스터를 보면 이 부분은 Name을 비교하는 것을 볼 수 있다.

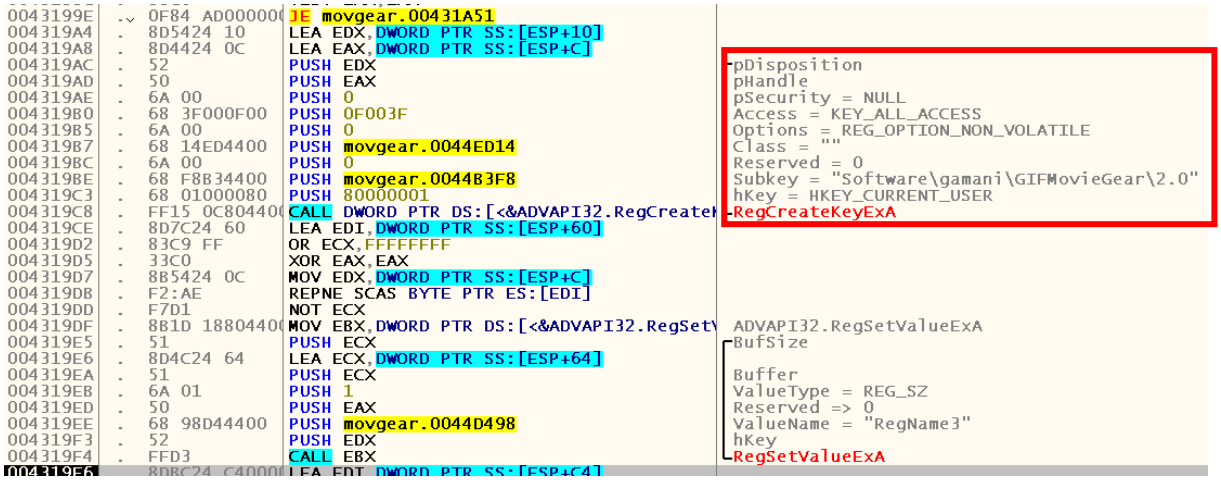


Name부분까지 실행해보면 0x00431636에서 EAX = 0인 곳으로 넘어가면 실패하는 곳으로 넘어가는 것을 볼 수 있다.

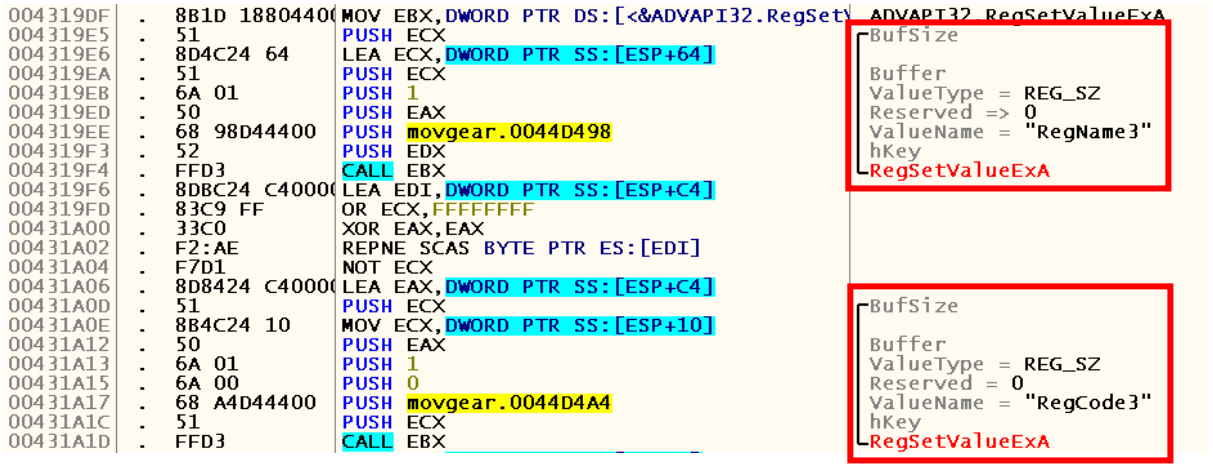
그럼 이 부분으로 넘어가지 않기 위해서는 JNZ분기에서 점프를 하지 않아야한다. ZF = 1로 바꿔서 실행한다.



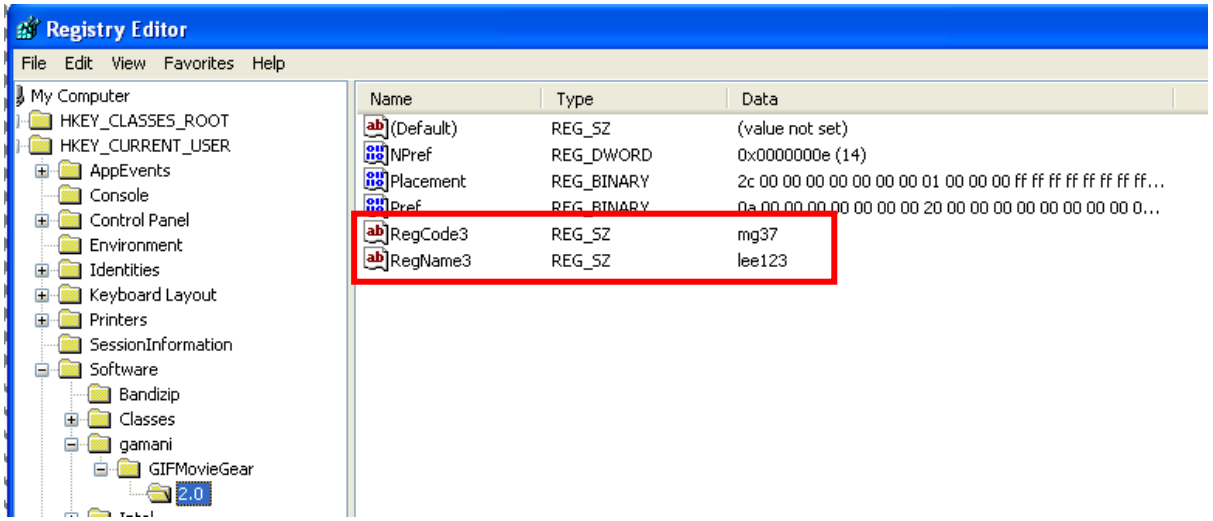
EAX = 1이여서 실패 창이 뜨는 곳(0x00431A5E)으로 점프하지 않고 실행되는 것을 볼 수 있다.



실행해보면 RegCreateKeyExA함수가 실행되는 것을 볼 수 있는데 이 함수를 찾아보면 '지정된 레지스트리 키를 만듭니다.'라고 나와있다. 아마 키를 여기다 저장한다는 거 같다.

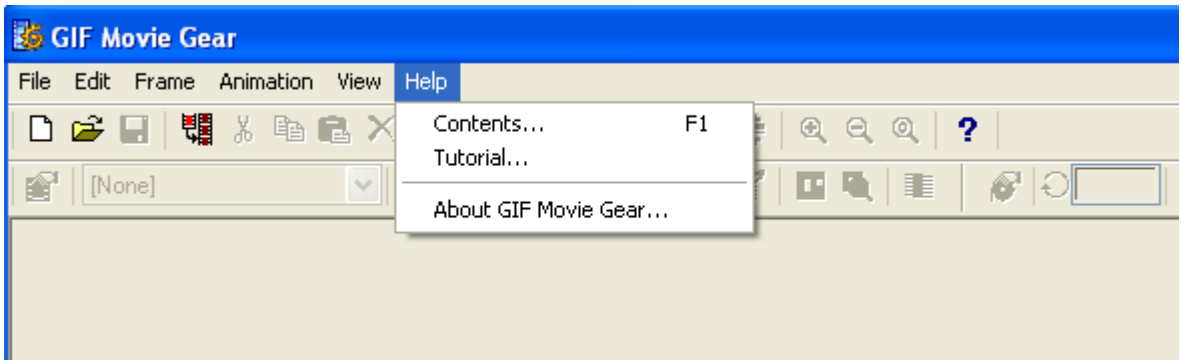


다시 실행해보면 RegSetValueExA함수가 실행되는 것을 볼 수 있는데 이 함수를 찾아보면 '열린 레지스트리 키와 연결된 지정된 값 이름의 형식 및 데이터를 검색'하는 함수라고 한다.

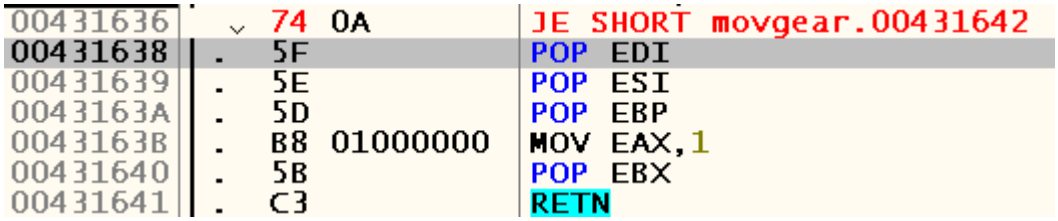


'win + r '로 regedit를 검색해서 RegCreateKeyExA에 나와있는 경로로 들어가보면 이렇게 name과 Code가 저장된 걸 볼 수 있다.

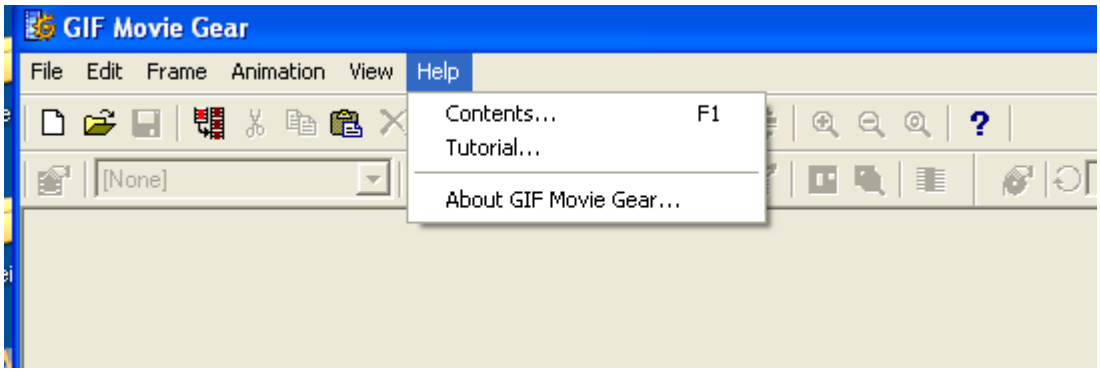
아마 처음에 name과 Code를 비교해서 맞으면 함수를 이용해서 레지스터에 저장하는 것 같다.



이제 끝까지 실행해보면 키 입력하는 부분이 없어진 걸 볼 수 있다.



0x00431636에 JNZ SHORT movgear.00431642 -> JE SHORT movgear.00431642로 바꿔서 저장해주고 실행해준다.



키 입력하는 부분도 없어지고 30일 남았다는 잔소리 창도 없어진 걸 볼 수 있다.

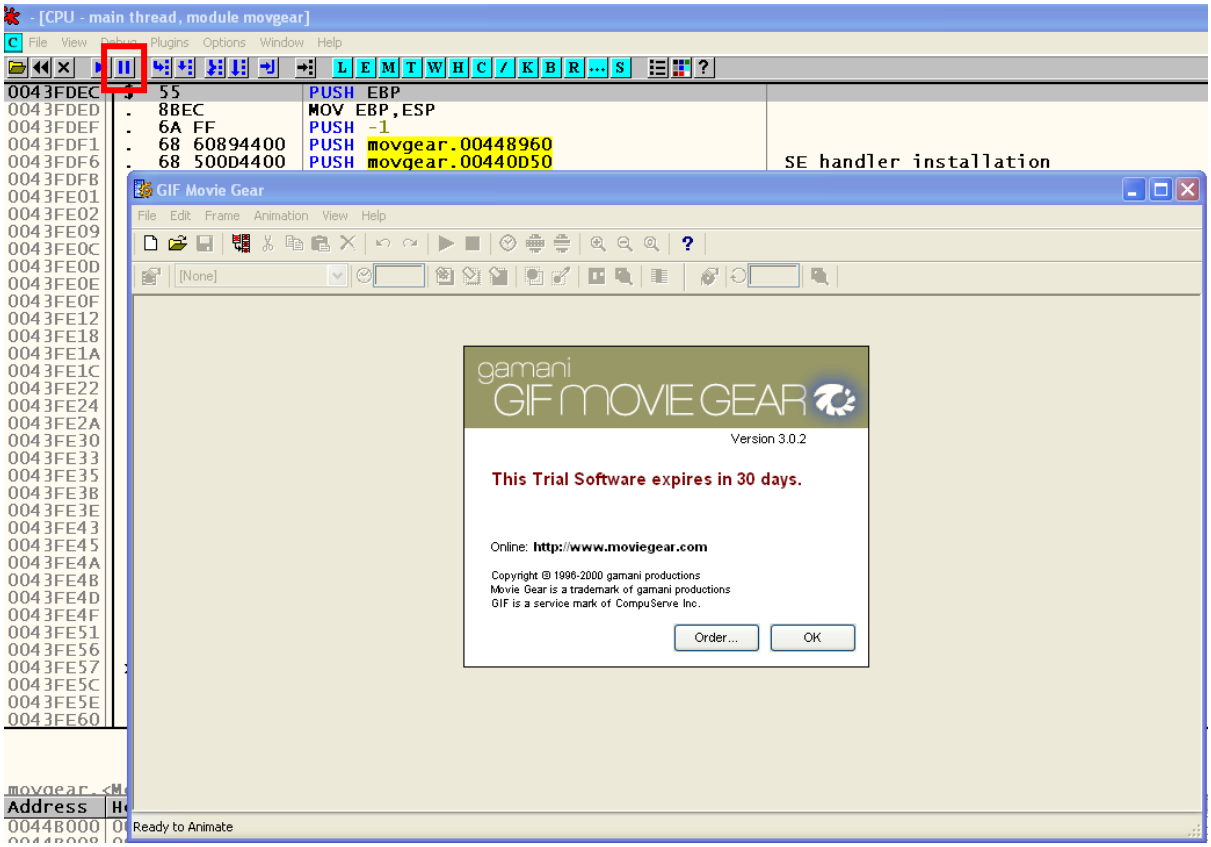
※ 저장하고 키 입력도 안했는데 입력하는 부분이 없어진 이유는 우리가 이걸 분석하면서 레지스터에 name과 code를 저장했기 때문에 없어진 것 같다.

문제 해결!

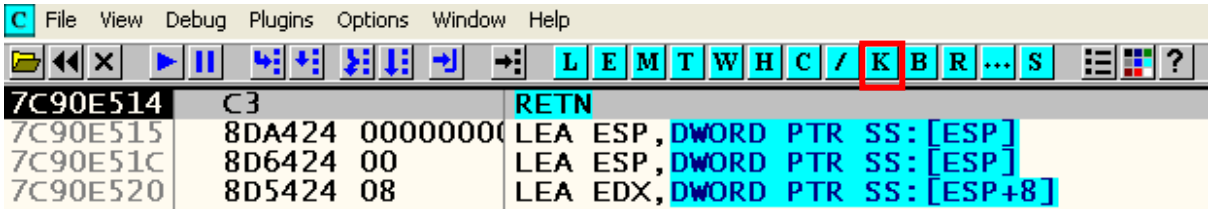
-> 하지만 lena의 영상을 보고 풀 경우 다른 방식으로 풀 수 있다. (근데 내가 좀 쉬움)

2-2. Lena 방식

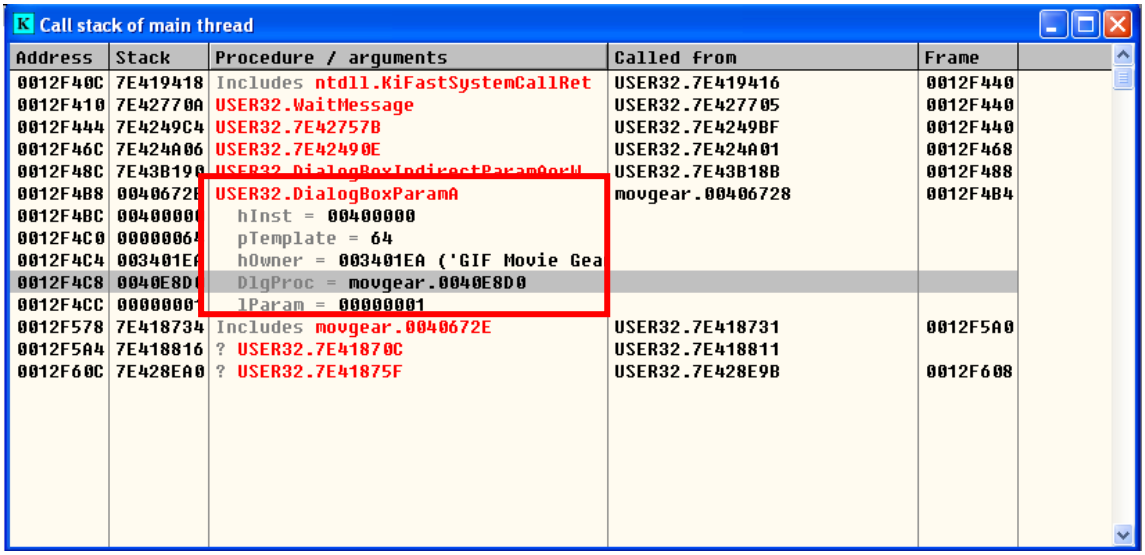
잔소리 창이 어디에 있는지 찾기 위해서 우선 실행을 하고 잔소리 창이 나올 때 멈춰주고 분석해본다.



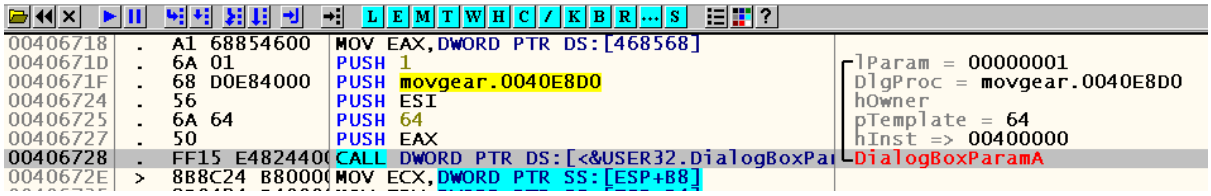
이렇게 잔소리 창이 나왔을 때 올리디버그로 돌아가서 멈춤버튼을 눌러준다.



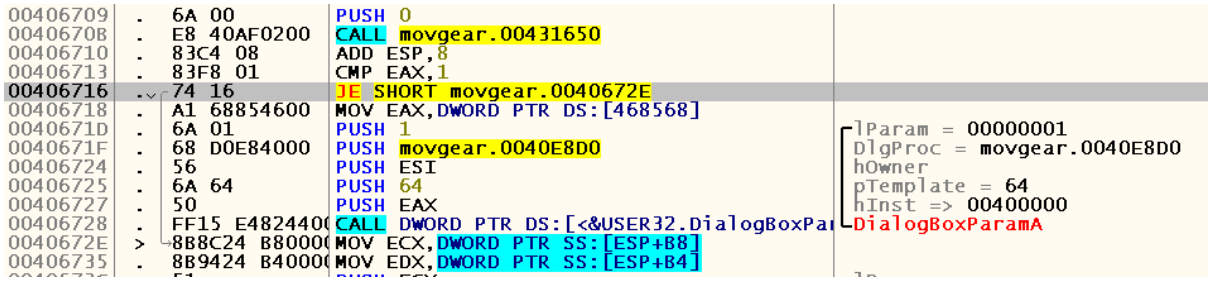
'K'버튼을 눌러준다. 지금까지 call한 함수를 차례로 볼 수 있는 창이다.



이 창에서 보면 잔소리 창을 발생시키는 함수는 DialogBoxParamA로 보인다.



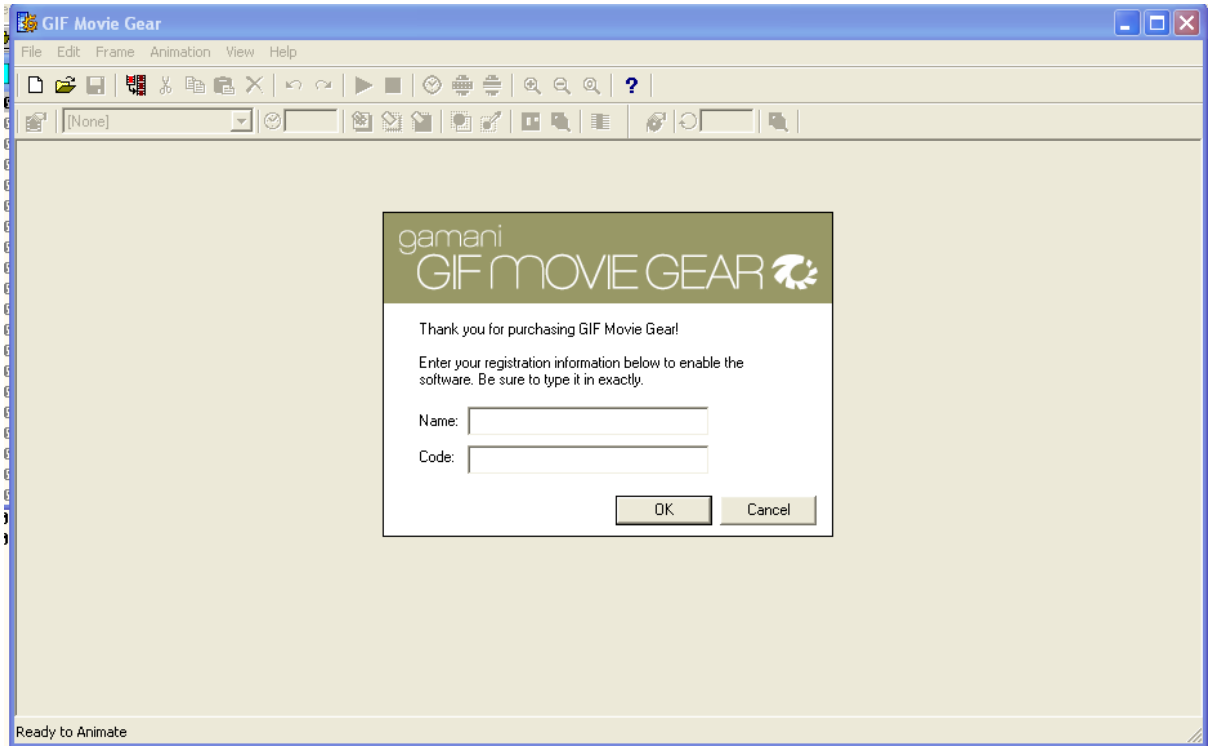
이 함수를 호출시킨 0x00406728를 눌러서 들어가면 해당 코드로 가는 걸 볼 수 있다.



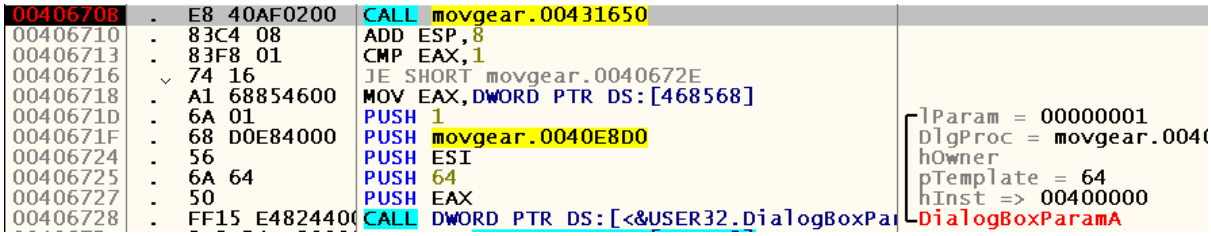
그럼 이 함수가 잔소리 창을 발생시키는 함수인데 주변 코드를 보면 이렇게 JE 분기문에서 해당 함수를 뛰어 넘을 수 있는 것을 볼 수 있다.

DoalogBoxParamA함수를 CMP와 관계 없이 JE -> JMP로 바꾸면 무조건 점프하기 때문에 JE SHORT movgear.0040672E -> JMP SHORT movgear.0040672E 로 변경하고 저장하고 실행해본다.

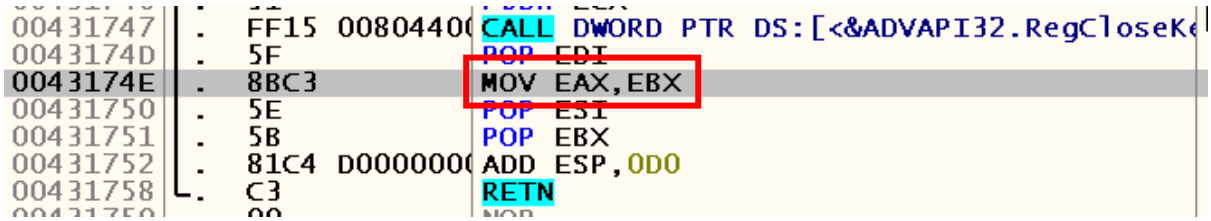
그럼 프로그램을 종료해도 잔소리 창이 안나오는 걸 볼 수 있다.



하지만 레지스터 키 입력하는 부분은 그대로 나온다. 그럼 이 방법은 틀린 방법이므로 다른 방법을 찾아야한다. 원래 파일을 이용해서 다시 풀어야한다.



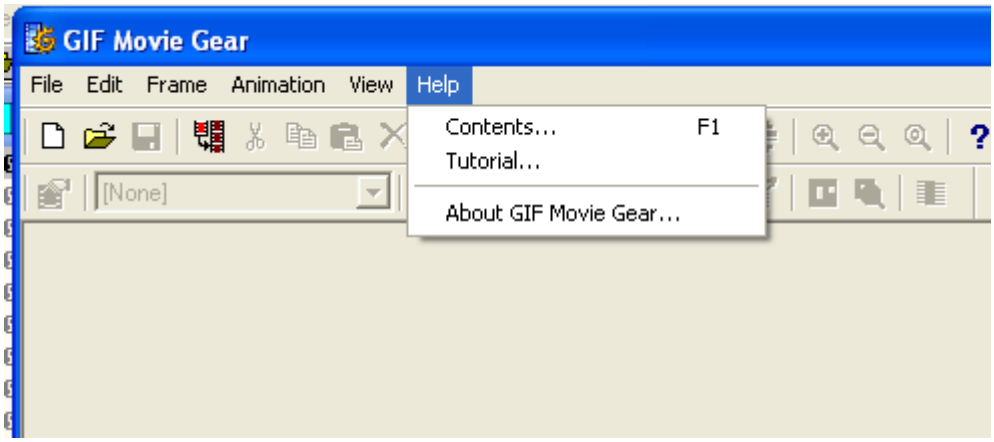
JMP로 바꾸는 것이 아닌 EAX = 1로 바꿔줘야한다. 0x0040670B를 'F7'을 눌러서 들어가서 EAX가 바뀌는 부분을 return함수 전부터 찾아준다.



0x0043174E에서 바뀌는 걸 알 수 있다. 그럼 이 부분을 바꿔줘야하는데 MOV EAX, EBX는 2byte인 걸 볼 수 있다.

근데 MOV EAX, 1로 바꾸면 8byte이므로 인라인패치를 해주거나 MOV AL, 1 (2byte)로 바

꿔주면 된다. MOV AL, 1로 바꿔서 저장하고 실행해준다.



그럼 잔소리창하고 키 등록하는 칸이 없어진 걸 확인할 수 있다.