

악성코드 분석 보고서

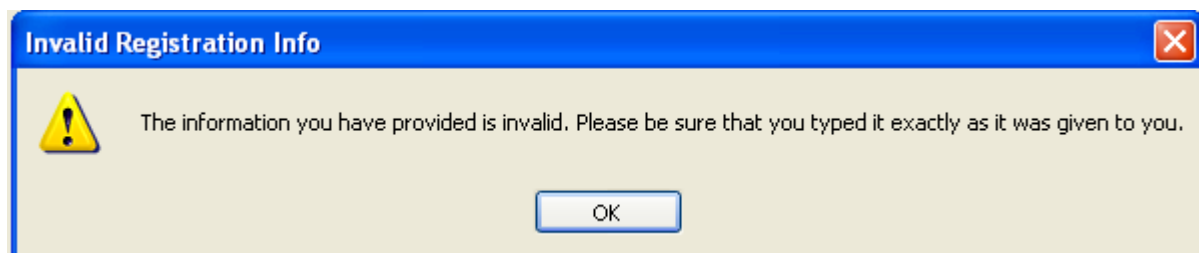
(sand-reversingwithlana-tutorials)

2025.07.21

1. 문제



레지스트리 키 등록

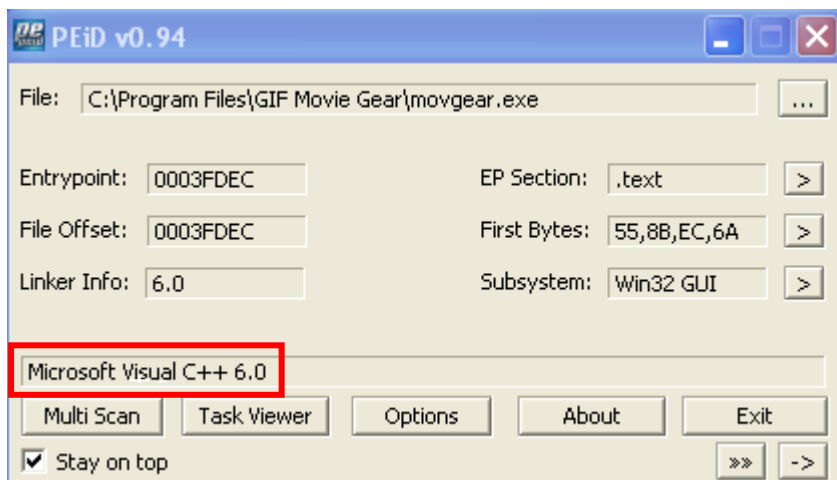


키 입력 틀렸을 때 나오는 구문



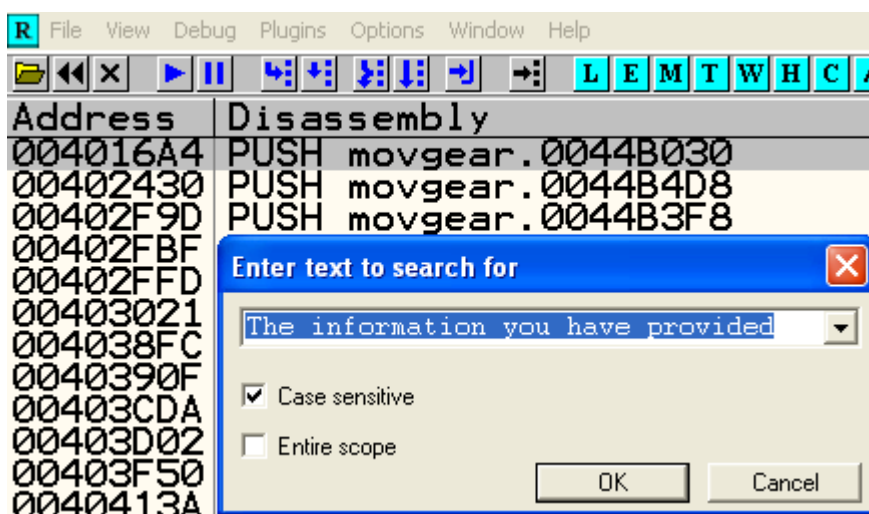
프로그램 나갈 때 나오는 잔소리 삭제하기 -> 아마 키를 등록하면 없어질거로 예상

2. 해결 방법

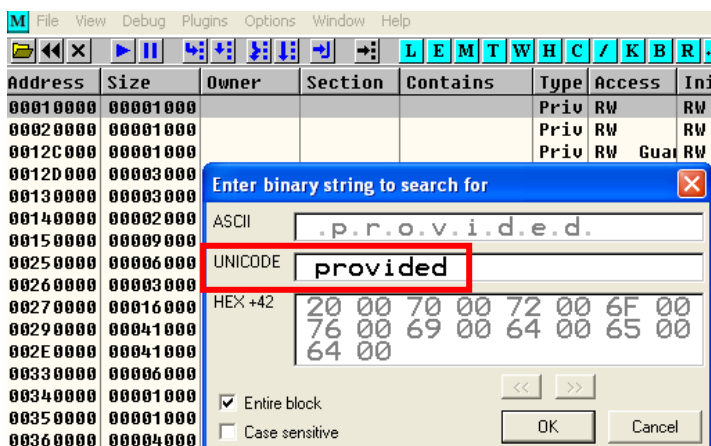


PEID로 확인 시 C++로 만든 파일이라는 것을 알 수 있다.

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.



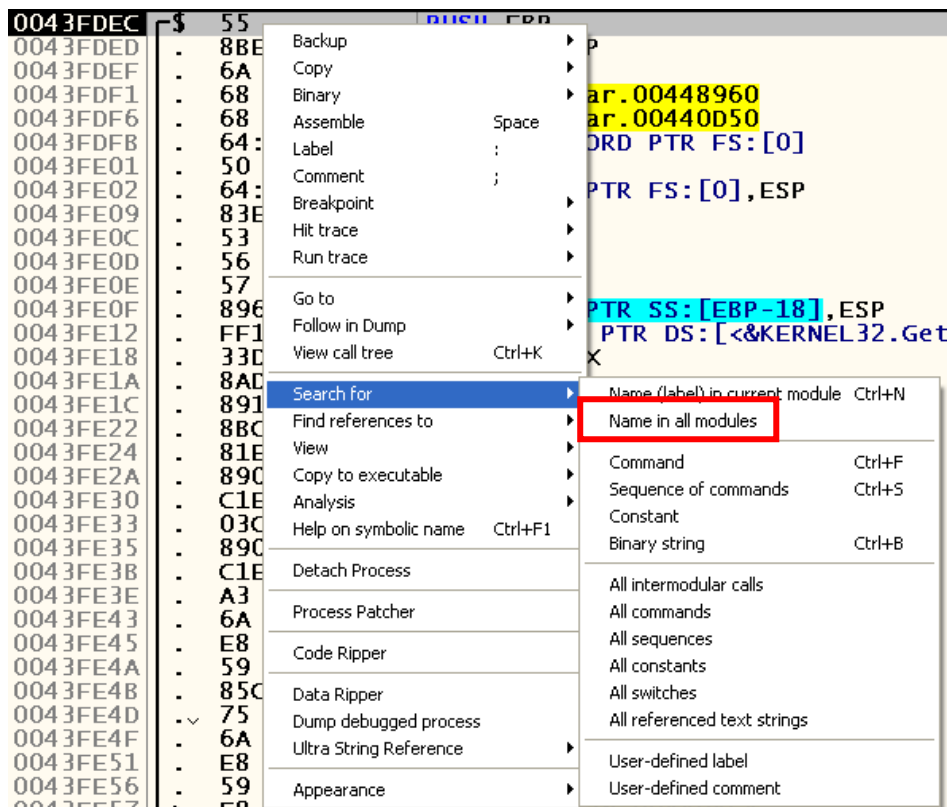
D Dump - movgear:.rsrc 0046A000..0048BFFF																	
004871D4	54	00	68	00	65	00	20	00	69	00	6E	00	66	00	6F	00	T.h.e. .i.n.f.o.
004871E4	72	00	6D	00	61	00	74	00	69	00	6F	00	6E	00	20	00	r.m.a.t.i.o.n. .
004871F4	79	00	6F	00	75	00	20	00	68	00	61	00	76	00	65	00	y.o.u. .h.a.v.e.
00487204	20	00	70	00	72	00	6F	00	76	00	69	00	64	00	65	00	.p.r.o.v.i.d.e.
00487214	64	00	20	00	69	00	73	00	20	00	69	00	6E	00	76	00	d. .i.s. .i.n.v.
00487224	61	00	6C	00	69	00	64	00	2E	00	20	00	50	00	6C	00	a.l.i.d... .P.l.
00487234	65	00	61	00	73	00	65	00	20	00	62	00	65	00	20	00	e.a.s.e. .b.e. .
00487244	73	00	75	00	72	00	65	00	20	00	74	00	68	00	61	00	s.u.r.e. .t.h.a.
00487254	74	00	20	00	79	00	6F	00	75	00	20	00	74	00	79	00	t. .y.o.u. .t.y.
00487264	70	00	65	00	64	00	20	00	69	00	74	00	20	00	65	00	p.e.d. .i.t. .e.
00487274	78	00	61	00	63	00	74	00	6C	00	79	00	20	00	61	00	x.a.c.t.l.y. .a.
00487284	73	00	20	00	69	00	74	00	20	00	77	00	61	00	73	00	s. .i.t. .w.a.s.
00487294	20	00	67	00	69	00	76	00	65	00	6E	00	20	00	74	00	.g.i.v.e.n. .t.
004872A4	6F	00	20	00	79	00	6F	00	75	00	2E	00	19	00	49	00	o. .y.o.u... .I.
004872B4	6F	00	76	00	61	00	6C	00	60	00	64	00	20	00	52	00	n u a l i d

Memory map에 들어가 UNICODE에 넣어보면 나오는 것을 볼 수 있다.

R References in movgear:.text to 004871D4..004871D4	
Address	Disassembly

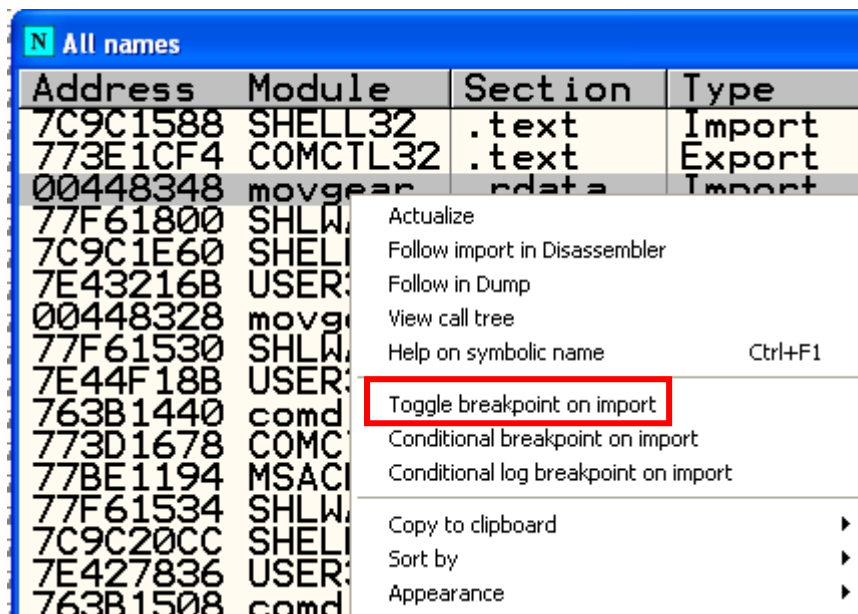
0x004871D4 메모리에 들어가서 참조하는 것을 찾아보면 아무것도 없는 걸 볼 수 있다.
그럼 이 방법이 아닌 다른 방법을 찾아야한다.

키를 입력하는 곳은 GetwindowTextA 함수를 사용했을 거라고 추정된다.



Find: GETWINDOWTEXT				
Address	Module	Section	Type	Name
7C9C1588	SHELL32	.text	Import	KERNEL32.GetWindowsDirec
773E1CE4	COMCTL32	.text	Export	GetWindowSubclass
00448348	movgear	.rdata	Import	USER32.GetWindowTextA
77FC1800	SHLWAPI	.text	Import	USER32.GetWindowTextA
7C9C1E60	SHELL32	.text	Import	USER32.GetWindowTextA
7E43216B	USER32	.text	Export	GetWindowTextA

총 4개가 있는데 그 중 모듈이 movgear에 브레이크를 걸어준다.



※ toggle breakpoint on import & conditional breakpoint on import 차이

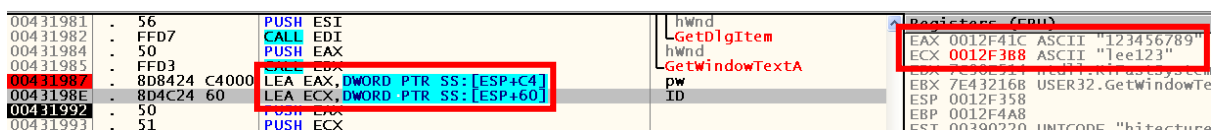
toggle breakpoint on import: 조건 없이 무조건 걸림

conditional breakpoint on import: 조건식을 만족하는 경우에만 멈춤.

이렇게 브레이크를 걸어주고 'F9'으로 실행해준다.



여기에 아무 값이나 넣고 실행해준다.



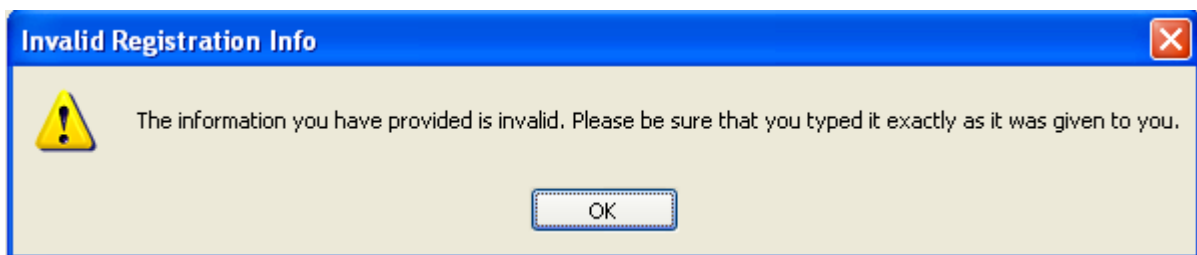
실행하다보면 레지스터에 code랑 name을 EAX와 ECX에 옮겨주는 코드를 볼 수 있다. 이 부분은 브레이크를 걸어주고 실행해준다.

00431993	.	51	PUSH ECX
00431994	.	E8 F7FBFFFF	CALL movgear.00431590
00431999	.	83C4 08	ADD ESP,8
0043199C	.	85C0	TEST EAX,EAX
0043199E	.	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	.	5C	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.

0043158E	.	90	NOP
0043158F	.	90	NOP
00431590	.	53	PUSH EBX
00431591	.	55	PUSH EBP
00431592	.	8B6C24 10	MOV EBP,DWORD PTR SS:[ESP+10]
00431596	.	56	PUSH ESI
00431597	.	57	PUSH EDI
00431598	.	807D 00 6D	CMP BYTE PTR SS:[EBP],6D
0043159C	.	0F85 A0000000	JNZ movgear.00431642
004315A2	.	807D 01 67	CMP BYTE PTR SS:[EBP+1],67
004315A6	.	0F85 96000000	JNZ movgear.00431642
004315AC	.	807D 02 33	CMP BYTE PTR SS:[EBP+2],33
004315B0	.	0F85 8C000000	JNZ movgear.00431642
004315B6	.	807D 03 37	CMP BYTE PTR SS:[EBP+3],37
004315BA	.	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

Registers (FPU)
EAX 0012F41C ASCII "123456789"
ECX 0012F3B8 ASCII "lee123"
EDX 7C90E514 ntdll.KiFastSystem
EBX 7E432168 USER32.GetWindowTe
ESP 0012F33C
EBP 0012F41C ASCII "123456789"
ESI 00390220 UNICODE "hitecture:
EDI 7E42436E USER32.GetDlgItem
EIP 0043159C movgear.0043159C
C 1 ES 0023 32bit 0(FFFFFFFF)
P 0 CS 001B 32bit 0(FFFFFFFF)
A 1 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 1 FS 003B 32bit 7FFDE000(FFF
T 0 GS 0000 NULL
D 0

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	. 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	. 7E 06	JLE SHORT movgear.00431625
0043161F	. 81EE BE170000	SUB ESI,17BE
00431625	> 83F9 0A	CMP ECX,0A
00431628	. 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	. 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	. 0F85 A0000000	JNZ movgear.00431642
004315A2	. 807D 01 67	CMP BYTE PTR SS:[EBP+1],67
004315A6	. 0F85 96000000	JNZ movgear.00431642
004315AC	. 807D 02 33	CMP BYTE PTR SS:[EBP+2],33
004315B0	. 0F85 8C000000	JNZ movgear.00431642
004315B6	. 807D 03 37	CMP BYTE PTR SS:[EBP+3],37
004315BA	. 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

0043160A	. 84D2	TEST DL,DL
0043160C	74 26	JF SHORT movgear.00431634
0043160E	0FBED2	MOV SX EDX,DL
00431611	41	INC ECX
00431612	0FAFD1	IMUL EDX,ECX
00431615	03F2	ADD ESI,EDX
00431617	81FE BE170000	CMP ESI,17BE
0043161D	7E 06	JLE SHORT movgear.00431625
0043161F	81EE BE170000	SUB ESI,17BE
00431625	83F9 0A	CMP ECX,0A
00431628	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	75 0A	JNZ SHORT movgear.00431642

레지스터를 보면 이 부분은 Name을 비교하는 것을 볼 수 있다.

00431636	75 0A	JNZ SHORT movgear.00431642
00431638	5F	POP EDI
00431639	5E	POP ESI
0043163A	5D	POP EBP
0043163B	B8 01000000	MOV EAX,1
00431640	5B	POP EBX
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

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 바뀌서 실행한다.

00431994	E8 F7FBFFFF	CALL movgear.00431590
00431999	83C4 08	ADD ESP,8
0043199C	85C0	TEST EAX,EAX
0043199E	0F84 AD000000	JF 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

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

0043199E	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	
004319AD	50	PUSH EAX	
004319AE	6A 00	PUSH 0	
004319B0	68 3F000F00	PUSH 0F003F	
004319B5	6A 00	PUSH 0	
004319B7	68 14ED4400	PUSH movgear.0044ED14	
004319BC	6A 00	PUSH 0	
004319BE	68 F8B34400	PUSH movgear.0044B3F8	
004319C3	68 01000080	PUSH 80000001	
004319C8	FF15 0C804400	CALL DWORD PTR DS:[<&ADVAPI32.RegCreateKeyExA]	
004319CE	8D7C24 60	LEA EDI, DWORD PTR SS:[ESP+60]	
004319D2	83C9 FF	OR ECX, FFFFFFFF	
004319D5	33C0	XOR EAX, EAX	
004319D7	8B5424 0C	MOV EDX, DWORD PTR SS:[ESP+C]	
004319DB	F2:AE	REPNE SCAS BYTE PTR ES:[EDI]	
004319DD	F7D1	NOT ECX	
004319DF	8B1D 18804400	MOV EBX, DWORD PTR DS:[<&ADVAPI32.RegSetValueExA]	
004319E5	51	PUSH ECX	
004319E6	8D4C24 64	LEA ECX, DWORD PTR SS:[ESP+64]	
004319EA	51	PUSH ECX	
004319EB	6A 01	PUSH 1	
004319ED	50	PUSH EAX	
004319EE	68 98D44400	PUSH movgear.0044D498	
004319F3	52	PUSH EDX	
004319F4	FFD3	CALL EBX	
004319F6	8DB8C24 C40000	LEA EDI, DWORD PTR SS:[ESP+C4]	
004319FD	83C9 FF	OR ECX, FFFFFFFF	
00431A00	33C0	XOR EAX, EAX	
00431A02	F2:AE	REPNE SCAS BYTE PTR ES:[EDI]	
00431A04	F7D1	NOT ECX	
00431A06	8D8424 C40000	LEA EAX, DWORD PTR SS:[ESP+C4]	
00431A0D	51	PUSH ECX	
00431A0E	8B4C24 10	MOV ECX, DWORD PTR SS:[ESP+10]	
00431A12	50	PUSH EAX	
00431A13	6A 01	PUSH 1	
00431A15	6A 00	PUSH 0	
00431A17	68 A4D44400	PUSH movgear.0044D4A4	
00431A1C	51	PUSH ECX	
00431A1D	FFD3	CALL EBX	

RegCreateKeyExA

pDisposition

pHandle

pSecurity = NULL

Access = KEY_ALL_ACCESS

Options = REG_OPTION_NON_VOLATILE

Class = ""

Reserved = 0

Subkey = "Software\gamani\GIFMovieGear\2.0"

hkey = HKEY_CURRENT_USER

ADVAPI32.RegSetValueExA

BufSize

Buffer

ValueType = REG_SZ

Reserved => 0

ValueName = "RegName3"

hKey

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

004319DF	8B1D 18804400	MOV EBX, DWORD PTR DS:[<&ADVAPI32.RegSetValueExA]	
004319E5	51	PUSH ECX	
004319E6	8D4C24 64	LEA ECX, DWORD PTR SS:[ESP+64]	
004319EA	51	PUSH ECX	
004319EB	6A 01	PUSH 1	
004319ED	50	PUSH EAX	
004319EE	68 98D44400	PUSH movgear.0044D498	
004319F3	52	PUSH EDX	
004319F4	FFD3	CALL EBX	
004319F6	8DB8C24 C40000	LEA EDI, DWORD PTR SS:[ESP+C4]	
004319FD	83C9 FF	OR ECX, FFFFFFFF	
00431A00	33C0	XOR EAX, EAX	
00431A02	F2:AE	REPNE SCAS BYTE PTR ES:[EDI]	
00431A04	F7D1	NOT ECX	
00431A06	8D8424 C40000	LEA EAX, DWORD PTR SS:[ESP+C4]	
00431A0D	51	PUSH ECX	
00431A0E	8B4C24 10	MOV ECX, DWORD PTR SS:[ESP+10]	
00431A12	50	PUSH EAX	
00431A13	6A 01	PUSH 1	
00431A15	6A 00	PUSH 0	
00431A17	68 A4D44400	PUSH movgear.0044D4A4	
00431A1C	51	PUSH ECX	
00431A1D	FFD3	CALL EBX	

ADVAPI32.RegSetValueExA

BufSize

Buffer

ValueType = REG_SZ

Reserved => 0

ValueName = "RegName3"

hKey

BufSize

Buffer

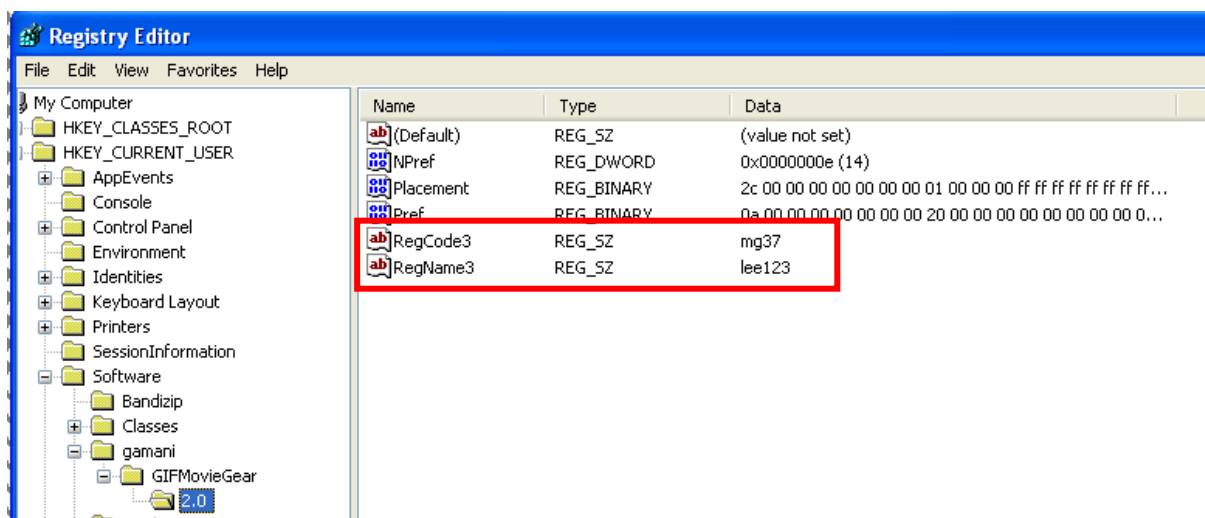
ValueType = REG_SZ

Reserved = 0

ValueName = "RegCode3"

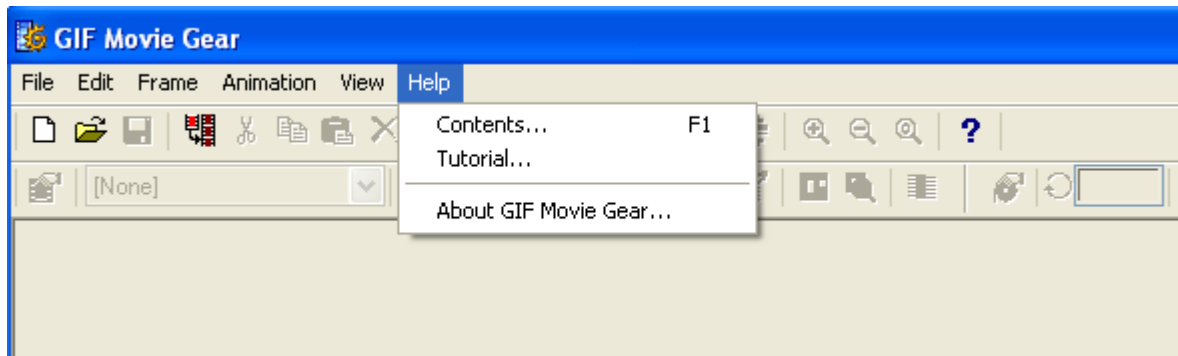
hKey

다시 실행해보면 RegSetValueExA 함수가 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.

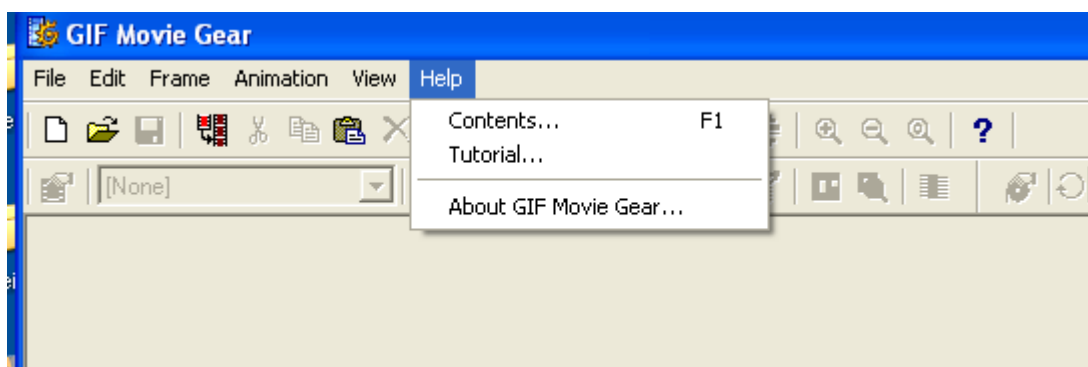
아마 처음에 name과 Code를 비교해서 맞으면 함수를 이용해서 레지스터에 저장하는 것 같다.



이제 끝까지 실행해보면 키 입력하는 부분이 없어진 걸 볼 수 있다.

00431636	74 0A	JE SHORT movgear.00431642
00431638	5F	POP EDI
00431639	5E	POP ESI
0043163A	5D	POP EBP
0043163B	B8 01000000	MOV EAX, 1
00431640	5B	POP EBX
00431641	C3	RETN

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



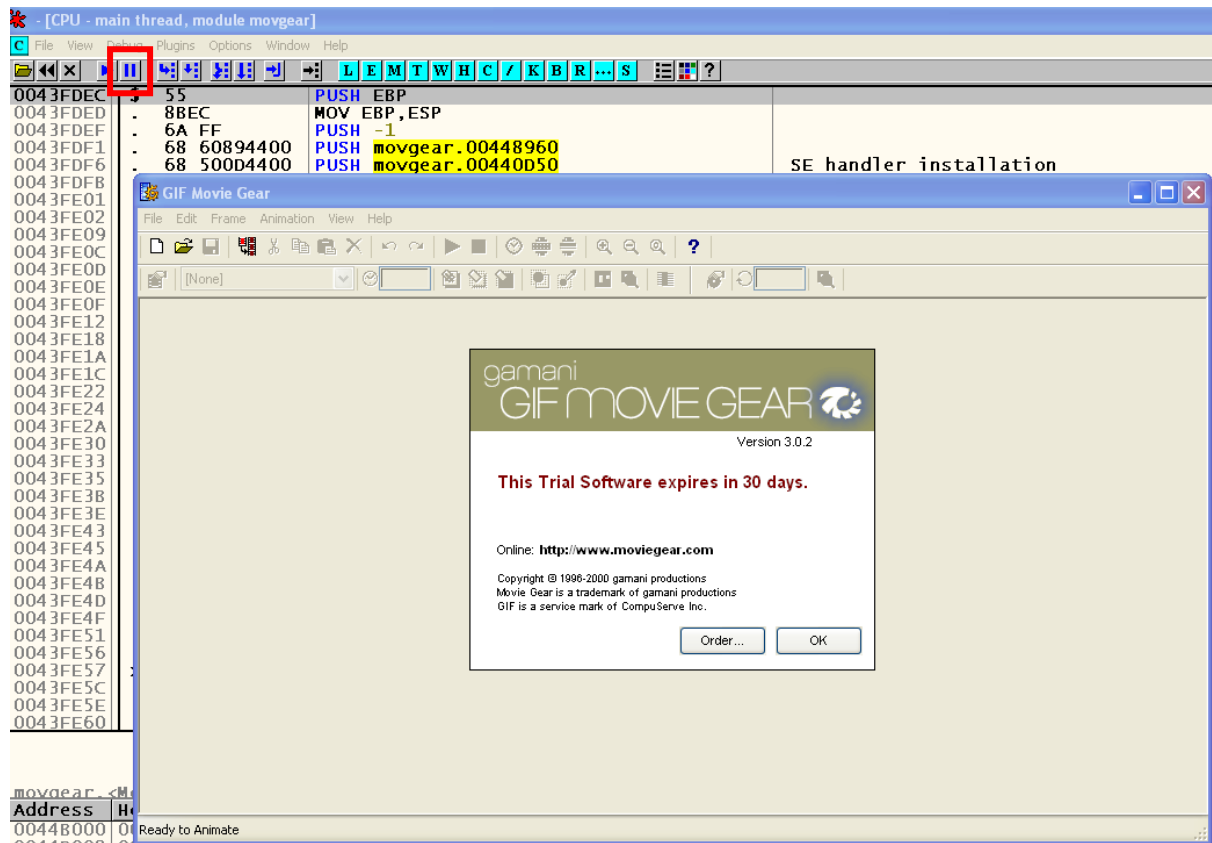
키 입력하는 부분도 없어지고 30일 남았다는 잔소리 창도 없어진 걸 볼 수 있다.

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

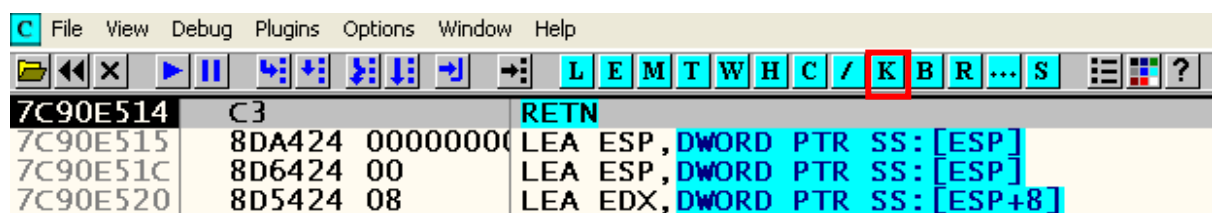
문제 해결!

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

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.

K Call stack of main thread					
Address	Stack	Procedure / arguments	Called from	Frame	
0012F40C	7E419418	Includes ntdll.KiFastSystemCallRet	USER32.7E419416	0012F440	
0012F410	7E42770A	USER32.WaitMessage	USER32.7E427705	0012F440	
0012F444	7E4249C4	USER32.7E42757B	USER32.7E4249BF	0012F440	
0012F46C	7E424A06	USER32.7E42490E	USER32.7E424A01	0012F468	
0012F48C	7E43B190	USER32.DialogBoxIndirectParam0orU	USER32.7E43B18B	0012F488	
0012F4B8	0040672E	USER32.DialogBoxParamA	movgear.00406728	0012F4B4	
0012F4BC	00400000	hInst = 00400000			
0012F4C0	00000064	pTemplate = 64			
0012F4C4	003401EA	hOwner = 003401EA ('GIF Movie Gea			
0012F4C8	0040E8D0	DlgProc = movgear.0040E8D0			
0012F4CC	00000001	lParam = 00000001			
0012F578	7E418734	Includes movgear.0040672E	USER32.7E418731	0012F5A0	
0012F5A4	7E418816	? USER32.7E41870C	USER32.7E418811		
0012F60C	7E428EA0	? USER32.7E41875F	USER32.7E428E9B	0012F608	

이 창에서 보면 잔소리 창을 발생시키는 함수는 DialogBoxParamA로 보인다.

L E M T W H C / K B R ... S					
00406718	. A1 68854600	MOV EAX,DWORD PTR DS:[468568]			
0040671D	. 6A 01	PUSH 1			
0040671F	. 68 D0E84000	PUSH movgear.0040E8D0			
00406724	. 56	PUSH ESI			
00406725	. 6A 64	PUSH 64			
00406727	. 50	PUSH EAX			
00406728	. FF15 E4824400	CALL DWORD PTR DS:[<&USER32.DialogBoxPar			
0040672E	> 8B8C24 B80000	MOV ECX,DWORD PTR SS:[ESP+B8]			

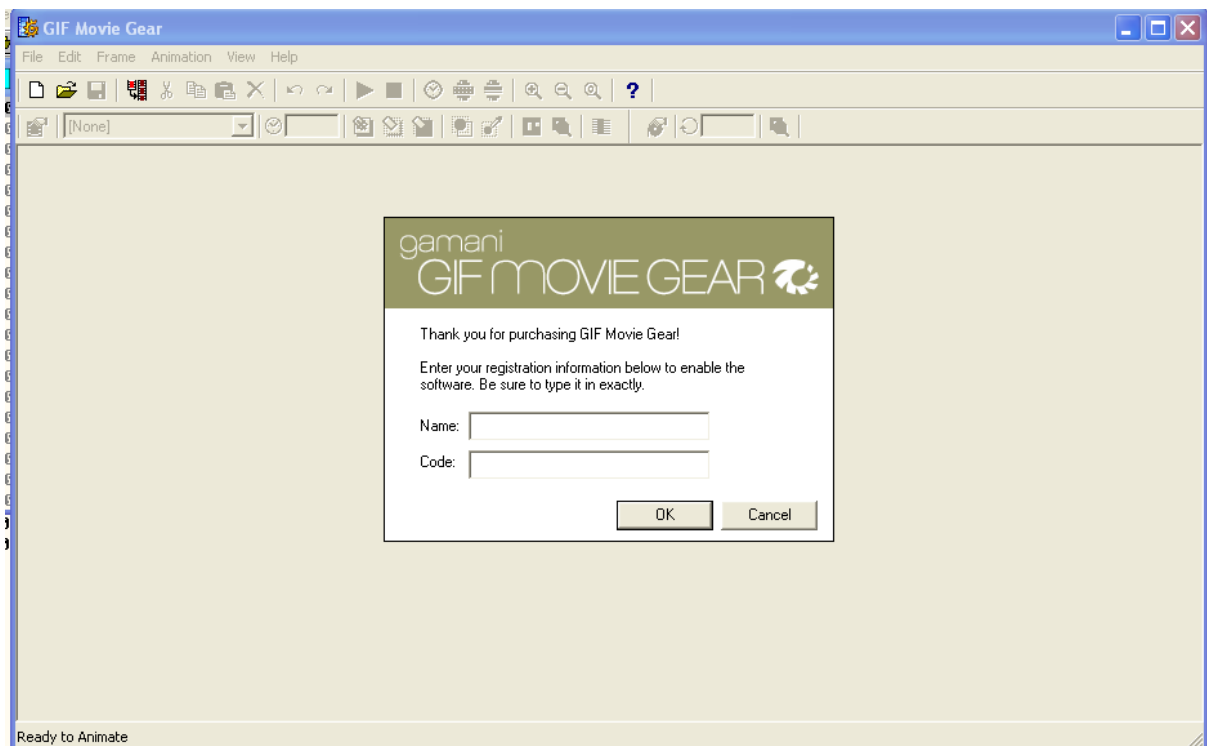
이 함수를 호출시킨 0x00406728를 눌러서 들어가면 해당 코드로 가는 걸 볼 수 있다.

00406709	. 6A 00	PUSH 0			
0040670B	. E8 40AF0200	CALL movgear.00431650			
00406710	. 83C4 08	ADD ESP,8			
00406713	. 83F8 01	CMP EAX,1			
00406716	. 74 16	JE SHORT movgear.0040672E			
00406718	. A1 68854600	MOV EAX,DWORD PTR DS:[468568]			
0040671D	. 6A 01	PUSH 1			
0040671F	. 68 D0E84000	PUSH movgear.0040E8D0			
00406724	. 56	PUSH ESI			
00406725	. 6A 64	PUSH 64			
00406727	. 50	PUSH EAX			
00406728	. FF15 E4824400	CALL DWORD PTR DS:[<&USER32.DialogBoxPar			
0040672E	> 8B8C24 B80000	MOV ECX,DWORD PTR SS:[ESP+B8]			
00406735	. 8B9424 B40000	MOV EDX,DWORD PTR SS:[ESP+B4]			

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.

0040670B	. E8 40AF0200	CALL movgear.00431650	
00406710	. 83C4 08	ADD ESP,8	
00406713	. 83F8 01	CMP EAX,1	
00406716	. 74 16	JE SHORT movgear.0040672E	
00406718	. A1 68854600	MOV EAX,DWORD PTR DS:[468568]	
0040671D	. 6A 01	PUSH 1	
0040671F	. 68 D0E84000	PUSH movgear.0040E8D0	
00406724	. 56	PUSH ESI	
00406725	. 6A 64	PUSH 64	
00406727	. 50	PUSH EAX	
00406728	. FF15 E4824400	CALL DWORD PTR DS:[<&USER32.DialogBoxParamA	DialogBoxParamA

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

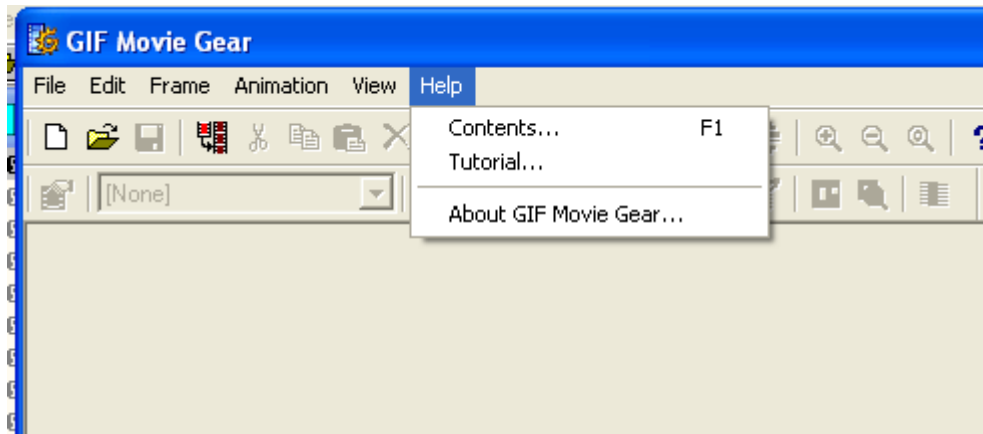
00431747	. FF15 00804400	CALL DWORD PTR DS:[<&ADVAPI32.RegCloseKey	
0043174D	. 5F	POP EDI	
0043174E	. 8BC3	MOV EAX,EBX	
00431750	. 5E	POP ESI	
00431751	. 5B	POP EBX	
00431752	. 81C4 D0000000	ADD ESP,0D0	
00431758	. C3	RETN	

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

B8 01000000	MOV EAX,1	B0 01	MOV AL,1
00	DB 00	00	DB 00

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

찍으면 된다. MOV AL, 1로 바꿔서 저장하고 실행해준다.



그럼 잔소리창하고 키 등록하는 칸이 없어진 걸 확인할 수 있다.