



프로그램 실행 시 성공 메시지가
뜬다. 이번엔 어떤 튜토리얼인지
디버거로 분석해본다.

00401103	. 43	RETN	
00401106	. 6A 00	PUSH 0	
00401108	. E8 AC010000	CALL <JMP.&kernel32.ExitProcess>	[ExitCode = 0 ExitProcess
0040110D	. C3	RETN	
0040110E	. \$ 6A 00	PUSH 0	
00401110	. 68 00204000	PUSH ReverseM.00402000	[Style = MB_OK!MB_APPLMODAL Title = "Key File ReverseMe" Text = "You really did it! Congrats !!!" hOwner = NULL MessageBoxA
00401115	. 68 51204000	PUSH ReverseM.00402051	
0040111A	. 6A 00	PUSH 0	[ExitProcess
0040111C	. E8 46020000	CALL <JMP.&user32.MessageBoxA>	
00401121	. E8 93010000	CALL <JMP.&kernel32.ExitProcess>	
00401126	. C3	RETN	

문자열로 검색해보니 아까 성공 메시지가 보이고, 해당 부분을 어디서 호출하는지 하나 하나 따라가보니 4010DD에서 호출한다.

00401000	> 43	INC EBX	
00401001	. ^EB EE	JMP SHORT ReverseM.004010C1	
00401003	> E8 23000000	CALL ReverseM.004010FB	
00401008	. 83FE 08	CMP ESI,8	
0040100B	. ^7C 05	JL SHORT ReverseM.004010E2	
0040100D	. E8 2C000000	CALL ReverseM.0040110E	
004010E2	> 6A 00	PUSH 0	
004010E4	. 68 00204000	PUSH ReverseM.00402000	[Style = MB_OK!MB_APPLMODAL Title = "Key File ReverseMe" Text = "Keyfile is not valid. Sorry." hOwner = NULL MessageBoxA
004010E9	. 68 86204000	PUSH ReverseM.00402086	
004010EE	. 6A 00	PUSH 0	[ExitProcess
004010F0	. E8 72020000	CALL <JMP.&user32.MessageBoxA>	
004010F5	. E8 BF010000	CALL <JMP.&kernel32.ExitProcess>	
004010FA	. C3	RETN	
004010FB	. \$ E8 D7010000	CALL <JMP.&kernel32.IsDebuggerPresent>	IsDebuggerPresent
00401100	. 83F8 01	CMP EAX,1	
00401103	. ^74 DD	JE SHORT ReverseM.004010E2	
00401105	. C3	RETN	
00401106	. 6A 00	PUSH 0	
00401108	. E8 AC010000	CALL <JMP.&kernel32.ExitProcess>	[ExitCode = 0 ExitProcess
0040110D	. C3	RETN	

해당 주소 위에 또 함수를 호출하고, 분기문이 있어서 보니 IsDebuggerPresent 결과에 따라 실행 결과가 달라진다. IsDebuggerPresent의 결과를 충족하면, 4010E2로 분기하는데, 보이는것과 같이 여기는 오류메시지가 뜨는 부분이다.

004010F5	. E8 BF010000	CALL <JMP.&kernel32.ExitProcess>	ExitProcess
004010FA	. C3	RETN	
004010FB	\$ E8 D7010000	CALL <JMP.&kernel32.IsDebuggerPresent>	IsDebuggerPresent
00401100	. 83F8 01	CMP EAX,1	
00401103	^74 DD	JE SHORT ReverseM.004010E2	
00401105	. C3	RETN	
00401106	. 6A 00	PUSH 0	ExitCode = 0
00401108	. E8 AC010000	CALL <JMP.&kernel32.ExitProcess>	ExitProcess
0040110D	. C3	RETN	
0040110E	\$ 6A 00	PUSH 0	Style = MB_OKIMB_F
00401110	. 68 00204000	PUSH ReverseM.00402000	Title = "Key File"

IsDebuggerPresent는 디버깅 중이면 1 아니면 0을 반환한다. 현재 디버깅중이니 1을 반환할거고, 그럼 오류메시지를 반환할 것이다.

004010E2	> 6A 00	PUSH 0	[Style = MB_OKIMB_APPLMODAL Title = "Key File ReverseMe" Text = "Keyfile is not valid. Sorry." hOwner = NULL MessageBoxA ExitProcess
004010E4	. 68 00204000	PUSH ReverseM.00402000	
004010E9	. 68 86204000	PUSH ReverseM.00402086	
004010EE	. 6A 00	PUSH 0	
004010F0	. E8 72020000	CALL <JMP.&user32.MessageBoxA>	MessageBoxA
004010F5	. E8 BF010000	CALL <JMP.&kernel32.ExitProcess>	ExitProcess
004010FA	. C3	RETN	
004010FB	\$ E8 D7010000	CALL <JMP.&kernel32.IsDebuggerPresent>	IsDebuggerPresent
00401100	. 83F8 01	CMP EAX,1	
00401103	<74 09	JE SHORT ReverseM.0040110E	
00401105	. C3	RETN	
00401106	. 6A 00	PUSH 0	ExitCode = 0

첫번째 방법으로 안티 디버깅에 걸리면 오류메시지로 분기하는 부분을 성공메시지로 분기하게 패치했다.

004010D0	> 43	INC EBX	
004010D1	^EB EE	JMP SHORT ReverseM.004010C1	
004010D3	> E8 23000000	CALL ReverseM.004010FB	
004010D8	. 83FE 08	CMP ESI,8	
004010DB	<7C 05	JL SHORT ReverseM.004010E2	
004010DD	. E8 2C000000	CALL ReverseM.0040110E	
004010E2	> 6A 00	PUSH 0	[
004010E4	. 68 00204000	PUSH ReverseM.00402000	
004010E9	. 68 86204000	PUSH ReverseM.00402086	
004010EE	. 6A 00	PUSH 0	
004010F0	. E8 72020000	CALL <JMP.&user32.MessageBoxA>	
004010F5	. E8 BF010000	CALL <JMP.&kernel32.ExitProcess>	
004010FA	. C3	RETN	
004010FB	\$ E8 D7010000	CALL <JMP.&kernel32.IsDebuggerPresent>	
00401100	. 83F8 00	CMP EAX,0	
00401103	^74 DD	JE SHORT ReverseM.004010E2	
00401105	. C3	RETN	

두번째 방법으로 디버깅 반환값과 비교하는 부분을 0으로 패치해 분기되지 않게 했다.

004010EE	. 6A 00	PUSH 0	hOwner = NULL
004010F0	. E8 72020000	CALL <JMP.&user32.MessageBoxA>	MessageBoxA
004010F5	. E8 BF010000	CALL <JMP.&kernel32.ExitProcess>	ExitProcess
004010FA	. C3	RETN	
004010FB	\$ E8 D7010000	CALL <JMP.&kernel32.IsDebuggerPresent>	IsDebuggerPresent
00401100	. 83F8 01	CMP EAX,1	
00401103	. 90	NOP	
00401104	. 90	NOP	
00401105	. C3	RETN	
00401106	. 6A 00	PUSH 0	ExitCode = 0
00401108	. E8 AC010000	CALL <JMP.&kernel32.ExitProcess>	ExitProcess
0040110D	. C3	RETN	

세번째 방법으로 분기되는 부분을 NOP으로 패치했다.