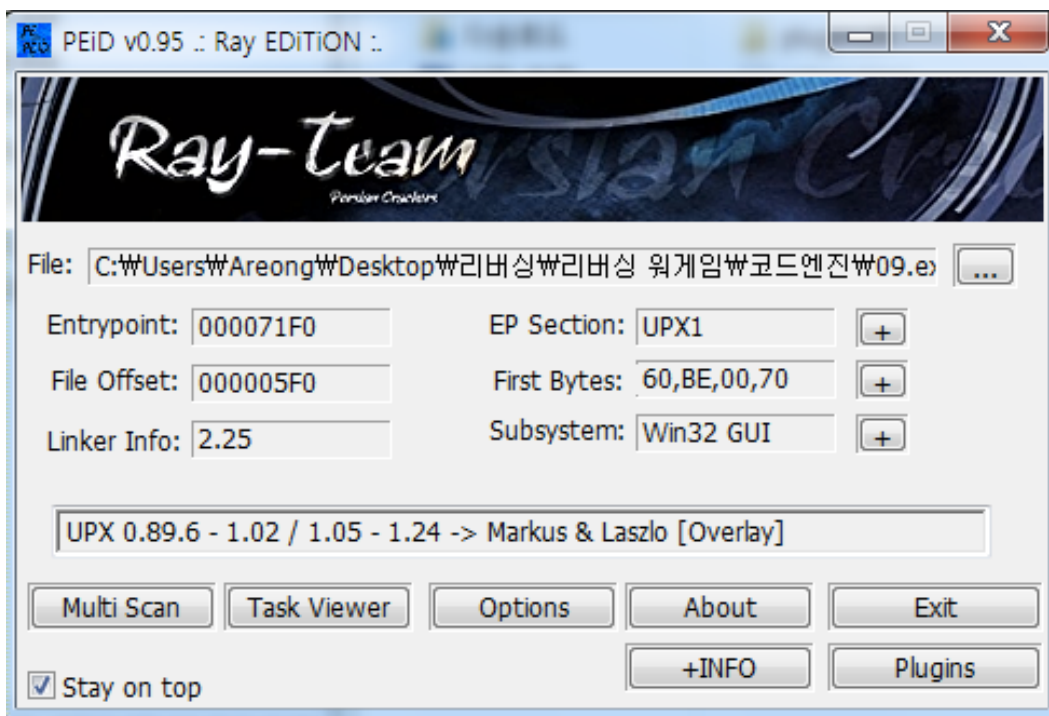
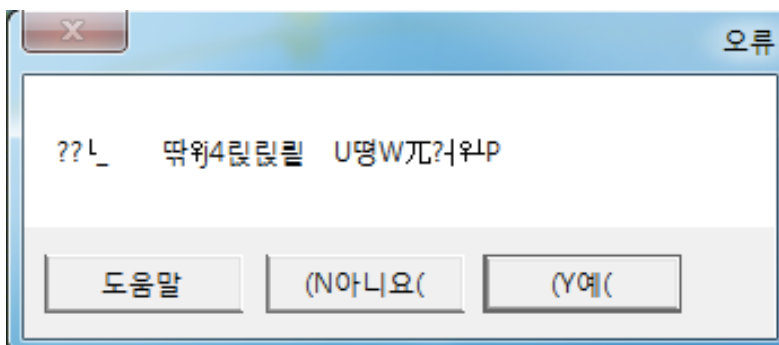


프로그램 실행 화면입니다. keyfile을 찾을 수 없다고 나옵니다.

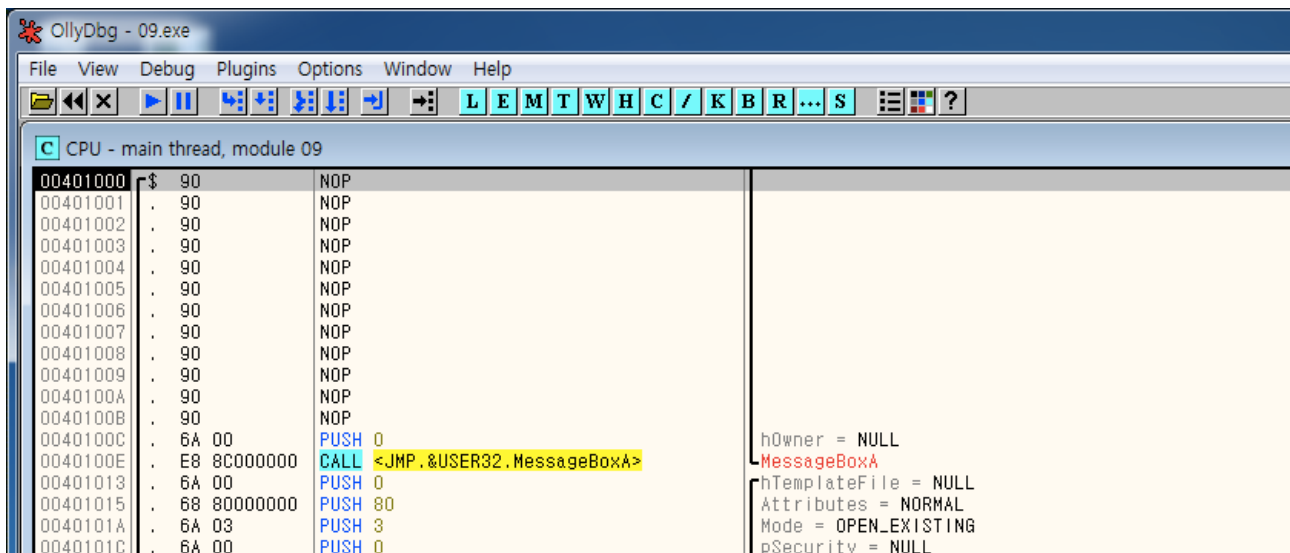


해당 프로그램은 UPX로 패킹되어 있음을 알 수 있습니다.

언패킹을 하고 진행하도록 하겠습니다.



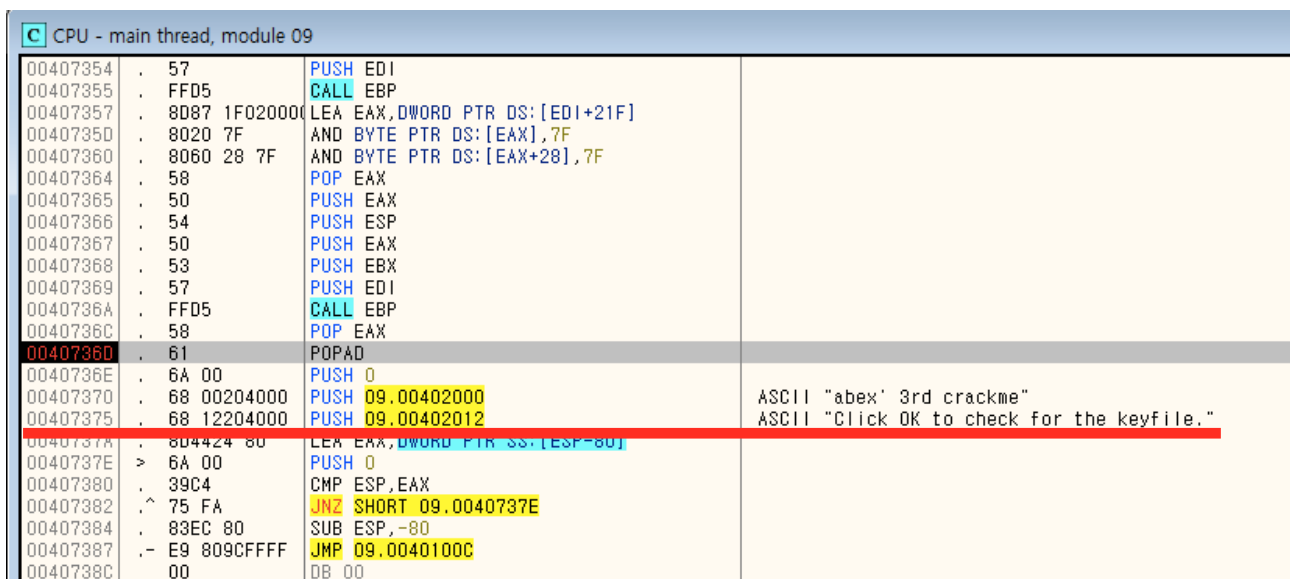
언패킹 후, 프로그램을 실행해보면 글자가 깨져있는 것을 알 수 있습니다.



이유가 무엇인지 올리디버거로 살펴보도록 하겠습니다.

MessageBox 함수를 호출하는 부분에 인자값 부분이 NOP으로 채워져 있습니다.

인자값이 NOP라서 메세지 박스의 글자가 깨지는 것을 알 수 있습니다.



언패킹을 하지 않은 코드입니다.

잘 보시면 POPAD뒤에 MessageBox의 인자값으로 보이는 문자열들이 있습니다.

00407367	. 50	PUSH EAX			EDI 00000000
00407368	. 53	PUSH EBX			EIP 00407387 09.00407387
00407369	. 57	PUSH EDI			C 1 ES 0023 32bit 0(FFFFFFFF)
0040736A	. FFD5	CALL EBP			P 0 CS 001B 32bit 0(FFFFFFFF)
0040736C	. 58	POP EAX			A 0 SS 0023 32bit 0(FFFFFFFF)
0040736E	. 6A 00	PUSH 0			Z 0 DS 0023 32bit 0(FFFFFFFF)
00407370	. 68 00204000	PUSH 09.00402000	ASCII "abex' 3rd crackme"		S 0 FS 003B 32bit 7FFDF000(FFF)
00407375	. 68 12204000	PUSH 09.00402012	ASCII "Click OK to check for the keyfile."		T 0 GS 0000 NULL
00407377	. 6A 00	PUSH 0			D 0
0040737E	. 39C4	CMP ESP,EAX			D 0 LastErr ERROR_SUCCESS (00000000)
00407382	. 75 FA	JNZ SHORT 09.0040737E			EFL 00000203 (NO,B,NE,BE,NS,PO,GE,G)
00407384	. 83EC 80	SUB ESP,-80			ST0 empty 0.0
00407387	. E9 809CFFFF	JMP 09.0040100C			ST1 empty 0.0
0040738C	. 00	DB 00			ST2 empty 0.0
0040738D	. 00	DB 00			ST3 empty 0.0
0040738E	. 00	DB 00			ST4 empty 0.0
0040738F	. 00	DB 00			ST5 empty 0.0
0040100C=09.0040100C					ST6 empty 1.00000000000000000000
					ST7 empty 1.1071487177940905030
					3 2 1 0 E S P U 0 Z 0 I
					FST 0020 Cond 0 0 0 0 Err 0 0 1 0 0 0 0
Address	Hex dump	ASCII			
00408000	00 00 00 00 DC 36 E5 37??			
00408008	00 00 00 00 00 00 01 00f.			
00408010	10 00 00 00 18 00 00 80	+...1..			
00408018	00 00 00 00 DC 36 E5 37??			
0012FF80	00402012	ASCII "Click OK to check for the keyfile."			
0012FF84	00402000	ASCII "abex' 3rd crackme"			
0012FF88	00000000				
0012FF8C	75F33C45	RETURN to kernel32.75F33C45			
0012FF90	7FFD4000				
0012FF94	7FFD4000				

그 문자열들이 JMP를 통해 원래 소스코드 흐름으로 돌아가기 전에 스택에 저장되어 있습니다.

0040100C	. 6A 00	PUSH 0			
0040100E	. E8 8C000000	CALL 09.0040109F	JMP to USER32.MessageBoxA		
00401013	. 6A 00	PUSH 0			
00401015	. 68 80000000	PUSH 80			
0040101A	. 6A 03	PUSH 3			
0040101C	. 6A 00	PUSH 0			
0040101E	. 6A 00	PUSH 0			
00401020	. 68 00000080	PUSH 80000000			
00401025	. 68 B9204000	PUSH 09.00402089	ASCII "abex.i2c"		
0040102A	. E8 5E000000	CALL 09.0040108D			
0040102F	. A3 CA204000	MOV DWORD PTR DS:[4020CA],EAX			
00401034	. 83F8 FF	CMP EAX,-1			
00401037	. 74 3C	JB SHORT 09.00401075			
00401039	. 6A 00	PUSH 0			
0040103B	. FF35 CA204000	PUSH DWORD PTR DS:[4020CA]			
00401041	. E8 40000000	CALL 09.00401093	JMP to kernel32.GetFileSize		
00401046	. 83F8 12	CMP EAX,12			
00401049	. 75 15	JNZ SHORT 09.00401060			
0040104B	. 6A 00	PUSH 0			
0040104D	. 68 35204000	PUSH 09.00402035	ASCII "Well done!"		
00401052	. 68 40204000	PUSH 09.00402040	ASCII "Yep, keyfile found!"		
00401057	. 6A 00	PUSH 0			
00401059	. E8 41000000	CALL 09.0040109F	JMP to USER32.MessageBoxA		
0040105E	. EB 28	JMP SHORT 09.00401088			
00401060	. 6A 00	PUSH 0			
00401062	. 68 79204000	PUSH 09.00402079	ASCII "Error"		
00401067	. 68 7F204000	PUSH 09.0040207F	ASCII "The found file is not a valid keyfile!"		
0040109F=09.0040109F					
Address	Hex dump	ASCII			
00408000	00 00 00 00 DC 36 E5 37??			
00408008	00 00 00 00 00 00 01 00f.			
00408010	10 00 00 00 18 00 00 80	+...1..			
00408018	00 00 00 00 DC 36 E5 37??			
0012FF7C	00000000	owner = NULL			
0012FF80	00402012	Text = "Click OK to check for the keyfile."			
0012FF84	00402000	Title = "abex' 3rd crackme"			
0012FF88	00000000	Style = MB_OK MB_APPLMODAL			
0012FF8C	75F33C45	RETURN to kernel32.75F33C45			
0012FF90	7FFD4000				

JMP한 후 바로 MessageBox를 호출하기 됩니다.

이처럼 특정 소스코드 라인을 숨겨놓는 방법이 Stolen byte라는 일종의 안티 디버깅 기법입니다.

원래의 흐름으로 돌아가기 전에 중요한 함수의 인자값을 스택에 저장하고 그 후 사용하게 됩니다.

이런 상태에서 툴이나 일반적인 방법으로 언패킹을 하게 되면 인자값을 가져오지 못하게 되는 경우가 발생합니다.