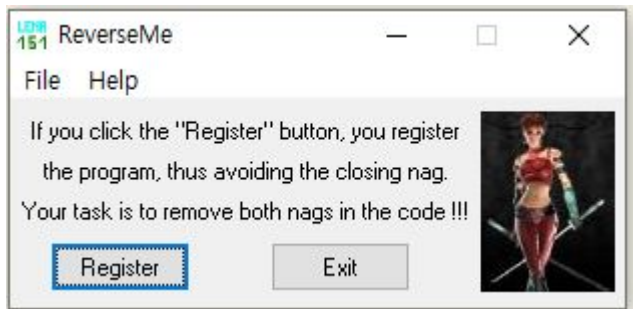


*m the starting na
Remove me*



*the closing
remove me to*



프로그램 실행 시 나타나는 Nag 창이다. 그리고 메인 프로그램이 실행되고, **Exit** 버튼을 누르면 또 다른 **Nag** 창이 나온다.

메시지를 해석해보면 둘다 삭제해달라고 한다.

디버깅을 통해 삭제하면 될것 같다.

0019BED0	75A03C55	win32u.NtUserGetMessage	USER32.75A03C4F	0019BF0C
0019BF10	6DC24425	USER32.GetMessageA	MFC42.6DC2441F	0019BF0C
0019BF14	004467CC	pMsg = ReverseM.004467CC		
0019BF18	00000000	hWnd = NULL		
0019BF1C	00000000	MsgFilterMin = 0		
0019BF20	00000000	MsgFilterMax = 0		
0019BF2C	6DC3E253	Includes MFC42.6DC24425	MFC42.6DC3E251	0019BF50
0019BF54	6DC50A0D	MFC42.#5718	MFC42.6DC50A08	0019BF50
0019BF9C	0042039F	? <JMP.&MFC42.#2514>	ReverseM.0042039A	0019BF98
0019FB84	0041D57E	? ReverseM.00420050	ReverseM.0041D579	
0019F9F8	0041A9F9	? ReverseM.0041D090	ReverseM.0041A9F4	
0019F9FC	00000016	Arg1 = 00000016		
0019FA00	0019FA30	Arg2 = 0019FA30		
0019FA74	0041A81F	ReverseM.0041A8D0	ReverseM.0041A81A	0019FA70
0019FAC4	0041A9A7	? ReverseM.0041A7C0	ReverseM.0041A9A2	0019FAC0
0019FB3C	0041A81F	ReverseM.0041A8D0	ReverseM.0041A81A	0019FB38
0019FB8C	0042CBF9	? ReverseM.0041A7C0	ReverseM.0042CBF4	0019FB88
0019FEAC	6DC24F3A	Includes ReverseM.0042CBF9	MFC42.6DC24F38	0019FEA8
0019FEC4	00433705	? <JMP.&MFC42.#1576>	ReverseM.00433700	0019FEC0
0019FED8	004332EC	? ReverseM.004336F0	ReverseM.<ModuleEntryPoint>	
0019FEDC	00400000	Arg1 = 00400000		
0019FEE0	00000000	Arg2 = 00000000		
0019FEE4	007F3975	Arg3 = 007F3975		
0019FEE8	0000000A	Arg4 = 0000000A		

시작 Nag 창을 분석하기 위해 실행하고, 디버깅 중지를 시켜 어디서 호출되는지 살펴본다.

Alt + K를 눌러 콜스택을 살펴본 결과 JMP.&MFC42.#2514부분에서 호출되는걸 알 수 있다.

해당 영역으로 가보니 MFC42.dll에서 제공하는 함수를 호출하는 루틴이 나온다. 그리고 그 위쪽을 살펴보니 분기문이 하나있고, EAX 값에 따라 해당 함수를 호출할지 말지를 결정한다.

00420377	. 85C0	TEST EAX,EAX
00420379	. v74 3F	JE SHORT ReverseM.004203BA
0042037B	. 8D4C24 4C	LEA ECX,DWORD PTR SS:[ESP+4C]
0042037F	. 89B424 CC3700	MOV DWORD PTR SS:[ESP+37CC],ESI
00420386	. 890D F4694400	MOV DWORD PTR DS:[4469F4],ECX
0042038C	. 8B56 08	MOV EDX,DWORD PTR DS:[ESI+8]
0042038F	. 8D4C24 4C	LEA ECX,DWORD PTR SS:[ESP+4C]
00420393	. 899424 A03700	MOV DWORD PTR SS:[ESP+37A0],EDX
0042039A	. E8 C1200100	CALL <JMP.&MFC42.#2514>
0042039F	. 8D8424 C83700	LEA EAX,DWORD PTR SS:[ESP+37C8]
004203A6	. 8D4C24 20	LEA ECX,DWORD PTR SS:[ESP+20]
004203AA	. 50	PUSH EAX
004203AB	. F8 2A270100	CALL <JMP.&MFC42.#858>

00420365	. 53	PUSH EBX
00420366	. 804C24 58	LEA ECX,DWORD PTR SS:[ESP+58]
0042036A	. C68424 1C3900	MOV BYTE PTR SS:[ESP+391C],3
00420372	. E8 C7290100	CALL <JMP.&MFC42.#3957>
00420377	. 85C0	TEST EAX,EAX
00420379	. 74 3F	JE SHORT ReverseH.004203BA
0042037B	. 804C24 4C	LEA ECX,DWORD PTR SS:[ESP+4C]
0042037F	. 89B424 CC3700	MOV DWORD PTR SS:[ESP+37CC],ESI
00420386	. 890D F4694400	MOV DWORD PTR DS:[4469F4],ECX
0042038C	. 8B56 08	MOV EDX,DWORD PTR DS:[ESI+8]
0042038F	. 804C24 4C	LEA ECX,DWORD PTR SS:[ESP+4C]
00420393	. 899424 A03700	MOV DWORD PTR SS:[ESP+37A0],EDX
0042039A	. E8 C1280100	CALL <JMP.&MFC42.#2514>
0042039F	. 808424 C83700	LEA EAX,DWORD PTR SS:[ESP+37C8]
004203A6	. 804C24 20	LEA ECX,DWORD PTR SS:[ESP+20]
004203AA	. 50	PUSH EAX
004203AB	. E8 2A270100	CALL <JMP.&MFC42.#858>
004203B0	. 8B4C24 1C	MOV ECX,DWORD PTR SS:[ESP+1C]

함수가 호출되기 전 분기문에 **bp**를
걸고 실행해본다. 그리고 분기가 되지
않아 밑에 함수를 호출한다. 그리고
계속 실행해보니 시작 **Nag**창을 띄우고
다시 분기문으로 이동하고, 똑같은
함수에서 메인 프로그램을 실행한다.
즉, **Nag** 창을 삭제하기 위해서 첫
분기에는 분기되게 하고, 두 번째는
분기되지 않게 하고, 세번째에는
분기되게 패치하면 될것 같다.

001A0000	00004000		Map	R	R	
001B0000	00003000		Map	R	R	
001C0000	00002000		Priv	RW	RW	
001D0000	00001000		Map	R	R	
001E0000	00004000		Map	R	R	
001F0000	00001000		Priv	RW	RW	
002E0000	00004000		Priv	RW	RW	
002E2000	00003000	data block	Priv	RW	RW	
002E5000	00003000	data block	Priv	RW	RW	
002E8000	00003000	data block	Priv	RW	RW	
002EB000	00003000	data block	Priv	RW	RW	
002EE000	00003000	data block	Priv	RW	RW	
002F1000	00003000	data block	Priv	RW	RW	
00274000	00004000	data block	Priv	RW	RW	
00400000	00001000	Reverse\$	PE header	Imag	R	RWE
00401000	000037000	Reverse\$	code	Imag	R	RWE
00438000	0000A000	Reverse\$.rdata	imports	Imag	R	RWE
00442000	00005000	Reverse\$.data	data	Imag	R	RWE
00447000	00005000	Reverse\$.rsro	resources	Imag	R	RWE
00450000	00006000		Map	R	R	
00460000	00001000		Map	RW	RW	
00470000	00003000		Priv	RW	RW	
00480000	00007000		Priv	RW	RW	
00490000	00003000		Map	R	R	
00550000	0000B000		Priv	RW	Gua: RW	
005D5000	0000B000		Priv	RW	Gua: RW	
00615000	0000B000		Priv	RW	Gua: RW	
00620000	00005000		Priv	RW	RW	

이번에도 전 튜토리얼과 같이
인라인 패치를 하면 될것 같다. JE를
만날 때마다. 특정 규칙에 의해
분기할지 말지를 결정하는 코드를
짜면 될것 같다.

445E70을 지역 변수 주소로 쓴다.
해당 주소에 있는 값과 어떤 값을
비교해 조건문에 만족할 경우에만
점프하도록 한다.(해당 주소를
어딘가에서 사용하고 있어 에러가
나서 **446AF0**으로 바꿨더니 됐다.)

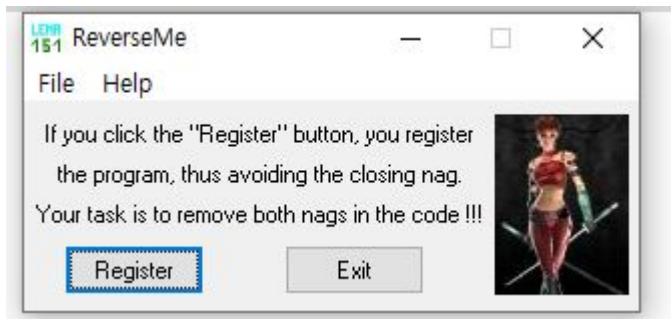
00437D62	00	DB 00
00437D63	00	DB 00
00437D64	00	DB 00
00437D65	00	DB 00
00437D66	00	DB 00
00437D67	FE05 705E4400	INC BYTE PTR DS:[445E70]
00437D6D	803D 705E4400	CMP BYTE PTR DS:[445E70],2
00437D74	0F85 40860A00	JNZ 004E03BA
00437D7A	8D4C24 4C	LEA ECX,DWORD PTR SS:[ESP+4C]
00437D7E	E9 FC85FEFF	JMP ReverseM.0042037F
00437D83	00	DB 00
00437D84	00	DB 00
00437D85	00	DB 00
00437D86	00	DB 00
00437D87	00	DB 00
00437D88	00	DB 00

이제 코드를 패치할 주소를 찾아야 한다. 그 전 튜토리얼과 마찬가지로 코드 맨 아래쪽으로 내리다 보면 안쓰는 공간이 있을것이다. 거기에 비교 코드를 삽입할 것이다.

이렇게 코드를 삽입한 후 원래 분기문을 인라인 패치한 부분으로 점프하도록 패치한다.

00420372	E8 C7290100	CALL <JMP.&MFC42.#3957>
00420377	85C0	TEST EAX,EAX
00420379	E9 E9790100	JMP ReverseM.00437D67
0042037E	90	NOP
0042037F	89B424 CC3700	MOV DWORD PTR SS:[ESP+37CC],ESI
00420386	890D F4694400	MOV DWORD PTR DS:[4469F4],ECX
0042038C	8B56 08	MOV EDI,DWORD PTR DS:[ESI+8]
0042038F	8D4C24 4C	LEA ECX,DWORD PTR SS:[ESP+4C]
00420393	899424 A03700	MOV DWORD PTR SS:[ESP+37A0],EDX
00420398	E8 C1280100	CALL <JMP.&MFC42.#2514>
0042039F	8D8424 C83700	LEA EAX,DWORD PTR SS:[ESP+37C8]
004203A6	8D4C24 20	LEA ECX,DWORD PTR SS:[ESP+20]
004203AA	50	PUSH EAX

그러면 445E70부분에서 2일 때만 메인 프로그램이 실행되고, 아닌 경우에는 모두 분기 될 것이므로 Nag창이 안뜰것이다.



이렇게 프로그램을 실행하면, Nag 창이 안뜨고 프로그램이 바로 실행된다.