

악성코드 분석 보고서

(sand-reversingwithlana-tutorials)

2025.06.07

1. 문제



체험판이라고 나오는 이 창을 없애야한다.



창을 다 닫았을 때 생기는 결제 유도 창을 없애야한다.

2. 해결 방법

“결과를 실행하는 함수”일 땐 **Called from**을 보고, 그걸 실행하게 만든 “판단 로직이 있을 함수”일 땐 **Procedure**를 눌러라!”

‘F8’ : 한 줄씩 내려가기

‘F8+ctrl’ : 한 줄씩 자동으로 내려가기

‘F9’ : 실행

‘F2’ : breakpoint 걸기

‘F7’ : 함수 내부 들어가서 보기

1) 체험판 창

004BD442	> 75 3A	JNZ SHORT VisualSi.004BD47E
004BD444	> 46	INC ESI
004BD445	. 8975 8C	MOV DWORD PTR SS:[EBP-74],ESI
004BD448	. 8A06	MOV AL,BYTE PTR DS:[ESI]
004BD44A	. 3AC3	CMP AL,BL
004BD44C	> 74 04	JE SHORT VisualSi.004BD452
004BD44E	. 3C 22	CMP AL,22
004BD450	^ 75 F2	JNZ SHORT VisualSi.004BD444
004BD452	> 803E 22	CMP BYTE PTR DS:[ESI],22
004BD455	> 75 04	JNZ SHORT VisualSi.004BD45B

‘F8+ctrl’를 눌러서 실행하는 것을 보면 이렇게 반복되는 구간을 볼 수 있다. 여기서 탈출하기 위해서는 그 다음 구문인 0x004BD452에 breakpoint를 걸어준다.

004BD444	> 46	INC ESI
004BD445	. 8975 8C	MOV DWORD PTR SS:[EBP-74],ESI
004BD448	. 8A06	MOV AL,BYTE PTR DS:[ESI]
004BD44A	. 3AC3	CMP AL,BL
004BD44C	> 74 04	JE SHORT VisualSi.004BD452
004BD44E	. 3C 22	CMP AL,22
004BD450	^ 75 F2	JNZ SHORT VisualSi.004BD444
004BD452	> 803E 22	CMP BYTE PTR DS:[ESI],22

걸고 ‘F9’를 누르면 빠르게 나올 수 있다.

‘F8’ 눌러서 실행하다보면 프로그램이 실행되는 구간이 있다.

004BD497	E8 74000000	CALL VisualSi.004BD510	VisualSi.004BD510
004BD49C	8945 98	MOV DWORD PTR SS:[EBP-68],EAX	
004BD49F	50	PUSH EAX	status
004BD4A0	FF15 2CCC4C00	CALL DWORD PTR DS:[<&MSVCRT.exit	exit
004BD4A6	8B45 EC	MOV EAX,DWORD PTR	
004BD4A9	8B08	MOV ECX,DWORD PTR	
004BD4AB	8B09	MOV ECX,DWORD PTR	
004BD4AD	894D 88	MOV DWORD PTR SS:[
004BD4B0	50	PUSH EAX	
004BD4B1	51	PUSH ECX	
004BD4B2	E8 27000000	CALL <JMP.&MSVCRT.	
004BD4B7	59	POP ECX	
004BD4B8	59	POP ECX	
004BD4B9	C3	RETN	
004BD4BA	8B	DB 8B	
004BD4BB	65	DB 65	
004BD4BC	E8	DB E8	
004BD4BD	FF	DB FF	
004BD4BE	75	DB 75	
004BD4BF	88FF	MOV BH,BH	
004BD4C1	15	DB 15	
004BD4C2	34CC4C00	DD <&MSVCRT.exit>	Number of trials left: 10

0x004BD497에서 실행되는 걸 볼 수 있다. 그럼 이 부분은 breakpoint를 걸어서 멈춰준다.

'F2+ctrl'을 누르고 재시작 후 'F9'을 눌러서 breakpoint가 걸린 0x004BD497로 와준다. 거기서 'F7'을 눌러 함수 내부를 본다. 또 'F8' 눌러서 실행하다보면 프로그램이 실행되는 구간이 또 있다.

004BD520	E8 43000000	CALL <JMP.&MFC42.#1576>	
004BD525	C2 1000	RETN 10	
004BD528	E8 B7F2FFFF	CALL <JMP.&MFC42.#	
004BD52D	8B4C24 04	MOV ECX,DWORD PTR	
004BD531	8B5424 08	MOV EDX,DWORD PTR	
004BD535	85C9	TEST ECX,ECX	
004BD537	8848 14	MOV BYTE PTR DS:[E	
004BD53A	8990 40100000	MOV DWORD PTR DS:[
004BD540	75 09	JNZ SHORT VisualSi	
004BD542	6A FD	PUSH -3	
004BD544	FF15 A8CB4C00	CALL DWORD PTR DS:	
004BD54A	59	POP ECX	
004BD54B	6A 01	PUSH 1	
004BD54D	58	POP EAX	
004BD54E	C2 0800	RETN 8	
004BD551	E9 00000000	JMP VisualSi.004BD	
004BD556	68 00060000	PUSH 600	
004BD55B	6A 00	PUSH 0	
004BD55D	E8 C6FFFFFF	CALL VisualSi.004B	
004BD562	A2 5C2D5900	MOV BYTE PTR DS:[5	Number of trials left: 10

0x004BD520에서 breakpoint를 걸어서 멈춰준다.

'F2+ctrl'을 누르고 재시작 후 'F9'을 눌러서 breakpoint가 걸린 0x004BD520로 와준다. 거기서 'F7'을 눌러 함수 내부를 본다. 또 'F8' 눌러서 실행하다보면 프로그램이 실행되는 구간이 또 있다.

CPU - main thread, module MFC42		
6A92350E	74 1C	JE SHORT MFC42.6A92352C
6A923510	8B16	MOV EDX,DWORD PTR DS:[ESI]
6A923512	8B42 58	MOV EAX,DWORD PTR DS:[EDX+58]
6A923515	8BCE	MOV ECX,ESI
6A923517	FFD0	CALL EAX

아까 실행하고 있던 주소랑 완전 다른 걸 볼 수 있다. 아까는 00으로 시작했고 지금은 6A로 시작하는 걸 볼 수 있다. 왜냐하면 MFC42.dll이라는 외부 함수로 들어왔기 때문이다. 그래서 주소가 완전 달라지고 **여기서는 Breakpoint 사용이 안된다.** (Hardware

breakpoint가 있긴한데 오류가 발생함.)

6BD43515	8BCE	MOV ECX,ESI	
6BD43517	FFD0	CALL EAX	VisualSi.00489310
6BD43519	8E60	TEST EAX,EAX	

그래서 breakpoint를 걸기 위해서는 0x00489310으로 들어가준다.

CPU - main thread, module VisualSi			
0048930E	90	NOP	
0048930F	90	NOP	
00489310	6A FF	PUSH -1	

0x00489310에서 breakpoint를 걸어주고 'F8'을 눌러 실행하다보면 프로그램이 실행되는 구간이 또 있다. (0x00489310에서는 굳이 breakpoint를 안걸어주고 실행해줘도 된다.)

CPU - main thread, module VisualSi			
004898F7	8D8C24 100200	LEA ECX,DWORD PTR SS:[ESP+210]	
004898FE	E8 6D240200	CALL VisualSi.004ABD70	
00489903	8D8C24 0C0200	LEA ECX,DWORD PTR SS:[ESP+20C]	
0048990A	C68424 788700	MOV BYTE PTR SS:[ESP+8778],0B	
00489912	E8 132B0300	CALL <JMP.&MFC42.#2514>	

0x00489912에서 breakpoint를 걸어주고 'F2+ctrl'을 누르고 재시작 후 'F9'을 눌러서 breakpoint가 걸린 0x00489912로 와준다. 거기서 'F7'을 눌러 함수 내부를 본다. 또 'F8'을 눌러서 실행하다보면 프로그램이 실행되는 구간이 또 있다.

CPU - main thread, module MFC42			
6BD71AC2	57	PUSH EDI	
6BD71AC3	8BCE	MOV ECX,ESI	
6BD71AC5	E8 B5D3FEFF	CALL MFC42.#5718	
6BD71ACA	837E 20 00	CMP DWORD PTR DS:[ESI+20],0	
6BD71ACE	74 16	JE SHORT MFC42.6BD71AE6	

이 부분은 breakpoint를 걸지 못하니까 다시 재시작을 해준 후 바로 'F7'을 눌러서 봐준다. 그리고 'F8'을 누르면 계속 반복하는 걸 볼 수 있다. **[K]**를 누르면 call stack을 볼 수 있다.

Call stack of main thread					
Address	Stack	Procedure / arguments	Called from	Frame	
00127704	6EA11AC4	? MFC42.#5718	MFC42.6EA11AC5	00127700	
00127708	00000004	Arg1 = 00000004			
00127748	00489917	? <JMP.&MFC42.#2514>	VisualSi.00489912	00127744	
0012FEC8	6E9E3519	Includes VisualSi.00489917	MFC42.6E9E3517		
0012FEDC	004BD525	? <JMP.&MFC42.#1576>	VisualSi.004BD520		
0012FEE0	00400000	Arg1 = 00400000			
0012FEE4	00000000	Arg2 = 00000000			
0012FEE8	00232428	Arg3 = 00232428			
0012FEEC	0000000A	Arg4 = 0000000A			
0012FEF0	004BD49C	? VisualSi.004BD510	VisualSi.<ModuleEntryPoint>		
0012FEF4	00400000	Arg1 = 00400000			
0012FEF8	00000000	Arg2 = 00000000			
0012FEFC	00232428	Arg3 = 00232428			
0012FF00	0000000A	Arg4 = 0000000A			

근데 우리는 체험판 창을 완전 없애는게 아니라 판단 로직을 이용해 수정하고 싶으니까 Procedure 쪽을 봐서 내부 함수가 있나 본다. 그럼 VisualSi.00489917 내부 함수 1개가 보인다. (지금 call stack 창은 위->아래로 쌓이기 때문에 위에서부터가 내가 최근에 실행한 함수이다. 만약 내부 함수가 여러 개 있으면 차례로 함수 코드 주변에 조건 분기분이 있나 봐준다.)

004898CC	. E8 AFF0FFFF	CALL VisualSi.00488980
004898D1	. 84C0	TEST AL,AL
004898D3	~ 0F84 FF000000	JE VisualSi.004899D8
004898D9	. 8A87 E0000000	MOV AL,BYTE PTR DS:[EDI+E0]
004898DF	. 84C0	TEST AL,AL
004898E1	~ 0F85 42010000	JNZ VisualSi.00489A29
004898E7	. 8B87 E4000000	MOV EAX,DWORD PTR DS:[EDI+E4]
004898ED	. 6A 00	PUSH 0
004898EF	. 85C0	TEST EAX,EAX
004898F1	~ 0F8E A1000000	JLE VisualSi.00489998
004898F7	. 8D8C24 100200	LEA ECX,DWORD PTR SS:[ESP+210]
004898FE	. E8 6D240200	CALL VisualSi.004ABD70
00489903	. 8D8C24 0C0200	LEA ECX,DWORD PTR SS:[ESP+20C]
0048990A	. C68424 788700	MOV BYTE PTR SS:[ESP+8778],0B
00489912	. E8 132B0300	CALL <JMP.&MFC42.#2514>

클릭해서 코드를 보면 내부 함수 위에 이렇게 3개가 있는 걸 볼 수 있다.

004898CC	. E8 AFF0FFFF	CALL VisualSi.00488980
004898D1	. 84C0	TEST AL,AL
004898D3	~ 0F84 FF000000	JE VisualSi.004899D8
004898D9	. 8A87 E0000000	MOV AL,BYTE PTR DS:[EDI+E0]
004898DF	. 84C0	TEST AL,AL
004898E1	~ 0F85 42010000	JNZ VisualSi.00489A29
004898E7	. 8B87 E4000000	MOV EAX,DWORD PTR DS:[EDI+E4]
004898ED	. 6A 00	PUSH 0
004898EF	. 85C0	TEST EAX,EAX
004898F1	~ 0F8E A1000000	JLE VisualSi.00489998
004898F7	. 8D8C24 100200	LEA ECX,DWORD PTR SS:[ESP+210]
004898FE	. E8 6D240200	CALL VisualSi.004ABD70
00489903	. 8D8C24 0C0200	LEA ECX,DWORD PTR SS:[ESP+20C]
0048990A	. C68424 788700	MOV BYTE PTR SS:[ESP+8778],0B
00489912	. E8 132B0300	CALL <JMP.&MFC42.#2514>

이렇게 3개 다 breakpoint를 걸어 주고 다시 실행해주고 비교해보면

JE (jump equal) : 비교 결과가 같을 때 점프

JNZ (jump not zero) : 결과가 0이 아닐 때 점프

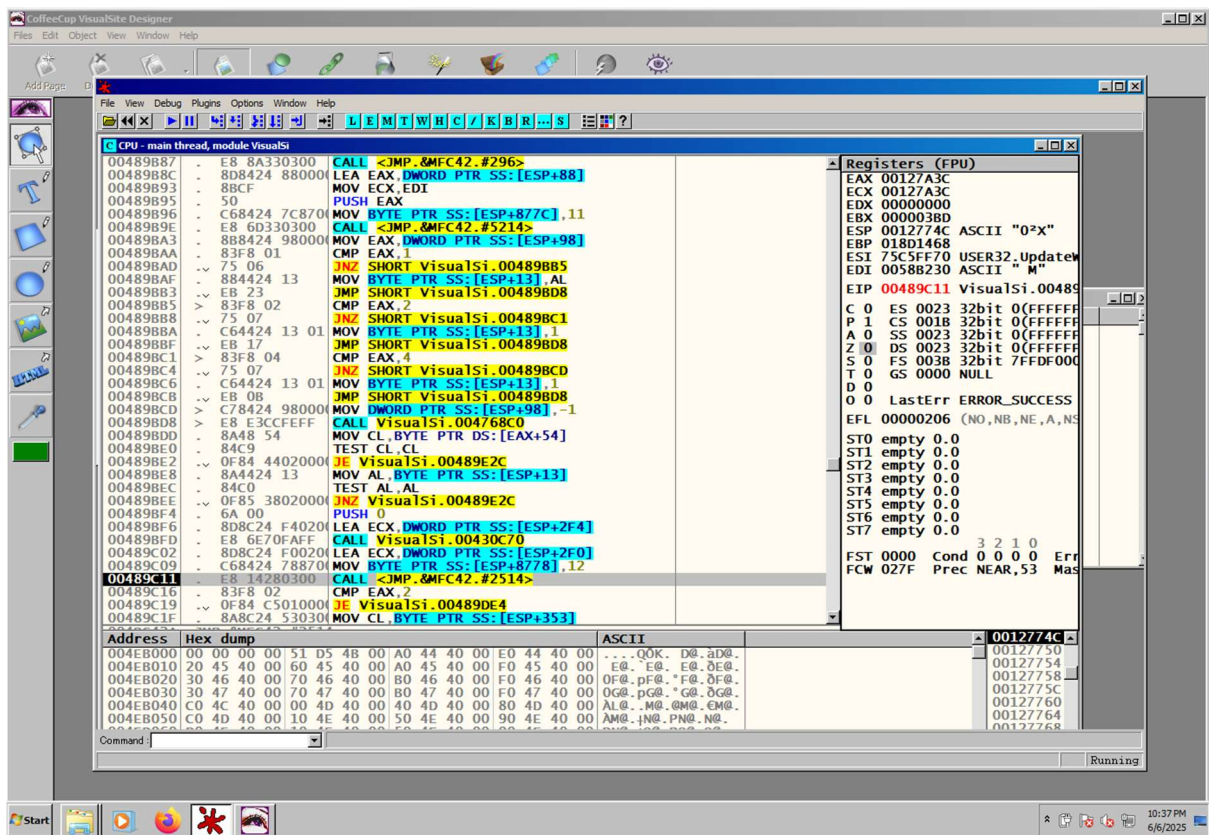
JLE (jump less or equal) : 작거나 같을 때 점프

첫 번째 JE를 보면 같으면 ZF=1이 되어야 점프한다. 그러면 ZF=1로 바꿔주고 'F8'로 실행해보면

CPU - main thread, module ntdll			
77846C74	C3	RETN	
77846C75	8DA424 00000000	LEA ESP,DWORD PTR SS:[ESP]	
77846C76	8B4224 00	LEA ECX,DWORD PTR SS:[ESP]	

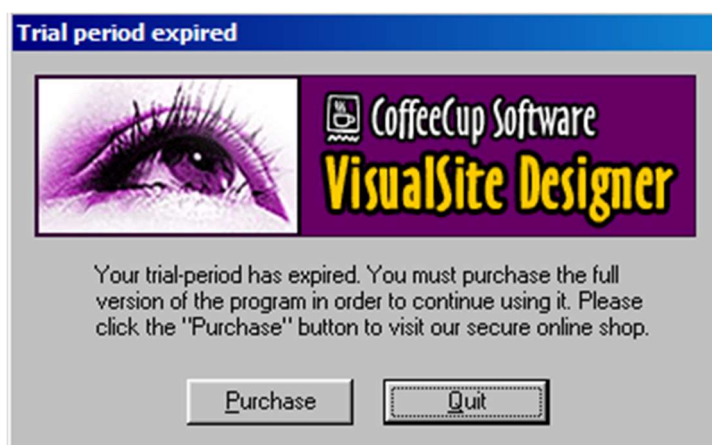
프로그램이 끝나는 걸 볼 수 있다. 그럼 이 부분은 바꾸면 안된다.

두 번째 JNZ를 보면 ZF=0이 되어야 점프를 한다. 그러면 ZF=0으로 바꾸고 'F8'로 실행해보면



뒤에 창이 실행되고 멈추는 걸 볼 수 있다. 그럼 이 코드를 수정해줘야한다.

세 번째 JLE를 보면 같으면 ZF=1이 되어야 점프한다. 그러면 ZF=1로 바꿔주고 'F8'로 실행해보면

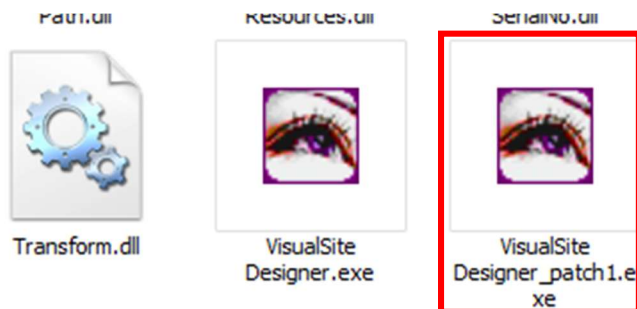


해당 창이 나오고 실행이 안되는 걸 볼 수 있다. 이 코드는 수정하면 안된다.

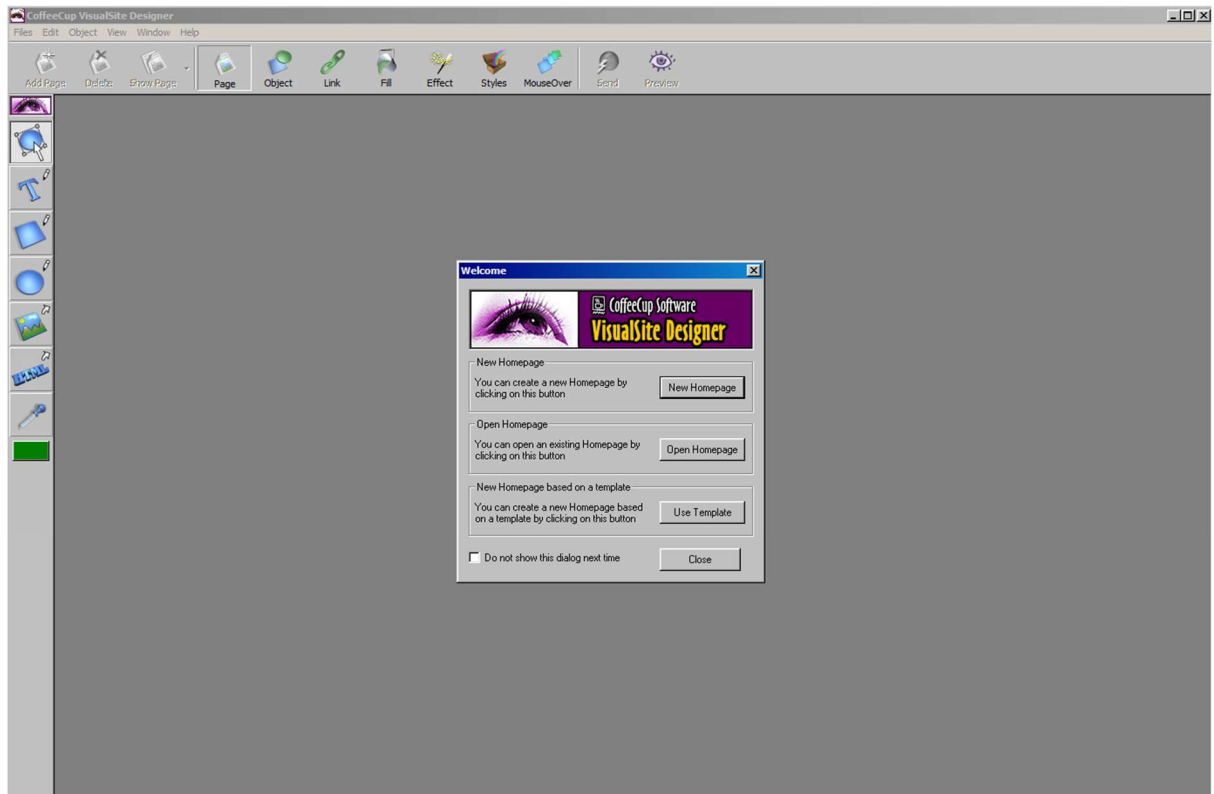
004898D1	.	84C0	TEST AL,AL
004898D3	✓	0F84 FF000000	JE VisualSi.004899D8
004898D9	.	8A87 E0000000	MOV AL,BYTE PTR DS:[EDI+E0]
004898DF	.	84C0	TEST AL,AL
004898E1	✓	E9 43010000	JMP VisualSi.00489A29
004898E6		008B 87E40000	ADD BYTE PTR DS:[EBX+E487],CL
004898EC	?	006A 00	ADD BYTE PTR DS:[EDX],CH
004898EF	.	85C0	TEST EAX,EAX
004898F1	✓	0F8E A1000000	JLE VisualSi.00489998
004898F7	.	8D8C24 100200	LEA ECX,DWORD PTR SS:[ESP+210]
004898FE	.	E8 6D240200	CALL VisualSi.004ABD70
00489903	.	8D8C24 0C0200	LEA ECX,DWORD PTR SS:[ESP+20C]
0048990A	.	C68424 788700	MOV BYTE PTR SS:[ESP+8778],0B
00489912	.	E8 132B0300	CALL <JMP.&MFC42.#2514>
00489917	.	83F8 01	CMP EAX,1

JNZ VisualSi.00489A29 -> JMP VisualSi.00489A29로 바꾸고 새로 저장해준다.

Copy to executable > Selection -> 덤프 창에서 Save File



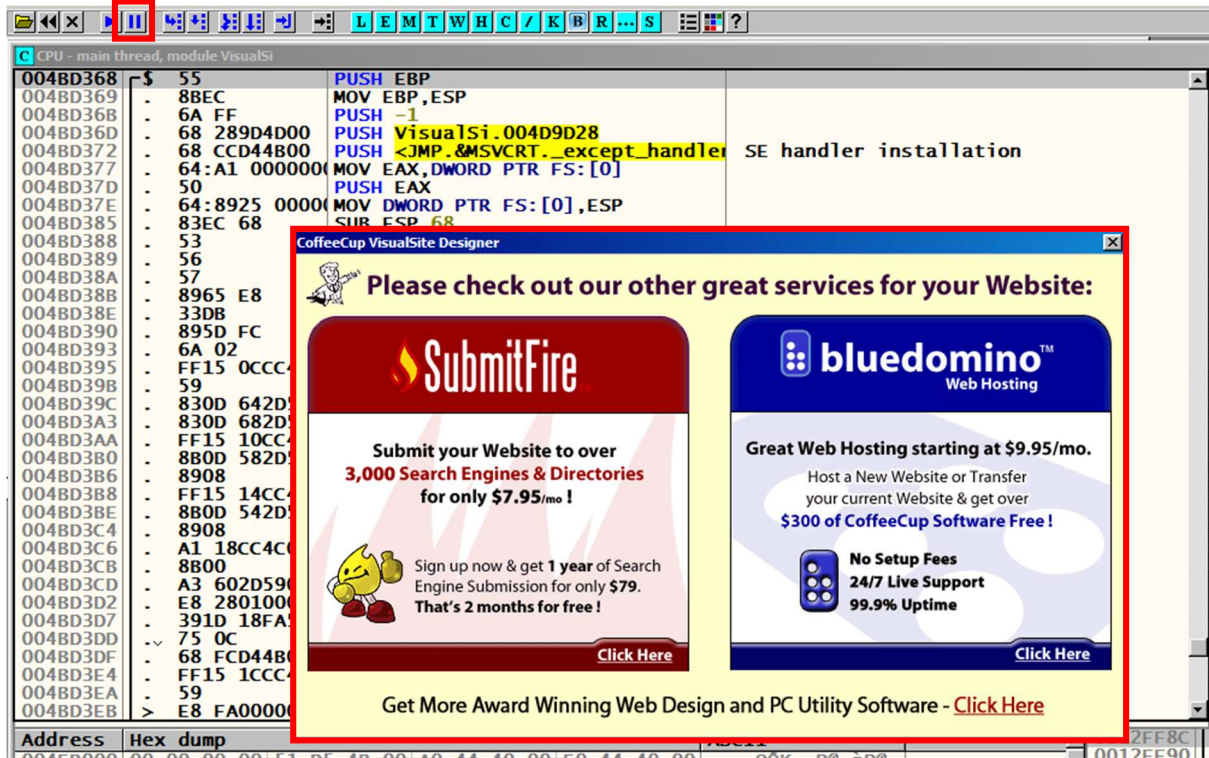
클릭해서 실행해보면



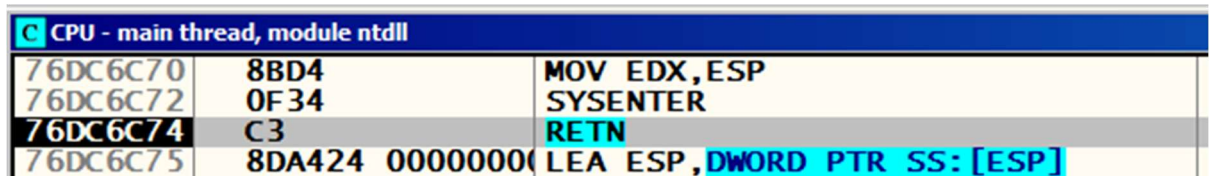
실행하는 걸 볼 수 있다.

2) 결제 유도 창

'F9'을 눌러서 프로그램을 실행시켜준다.



모든 창을 끄고 결제 유도창만 남은 채로 빨간 박스인 '멈춤' 버튼을 누르면



끝난 걸 볼 수 있다. 그 후 call stack창을 열어서 보면 실행된 내부 함수를 볼 수 있다.

Call stack of main thread					
Address	Stack	Procedure / arguments	Called from	Frame	
0012F094	769ECD80	Includes ntdll.KiFastSystemCallRet	USER32.769ECDAE	0012F0B8	
0012F098	769E18A9	USER32.769ECDA4	USER32.769E18A4	0012F0B8	
0012F0BC	6BAB541F	USER32.GetMessageA	MFC42.6BAB5419	0012F0B8	
0012F0C0	0058B264	pMsg = VisualSi.0058B264			
0012F0C4	00000000	hWnd = NULL			
0012F0C8	00000000	MsgFilterMin = 0			
0012F0CC	00000000	MsgFilterMax = 0			
0012F0D8	6BACEF51	Includes MFC42.6BAB541F	MFC42.6BACEF4F	0012F0F4	
0012F0F8	6BAE1ACA	MFC42.#5718	MFC42.6BAE1AC5	0012F0F4	
0012F13C	00480C29	? <JMP.&MFC42.#2514>	VisualSi.00480C24	0012F138	
0012F1AC	6BAAA5A1	Includes VisualSi.00480C29	MFC42.6BAAA59F	0012F228	
0012F22C	6BAA3687	Includes MFC42.6BAAA5A1	MFC42.6BAA3685	0012F228	
0012F254	6BAAA361	Includes MFC42.6BAA3687	MFC42.6BAAA35F	0012F250	
0012F2BC	6BAAA2B9	MFC42.#1109	MFC42.6BAAA2B4	0012F2B8	
0012F2E0	6BAAA571	MFC42.#1578	MFC42.6BAAA56C	0012F2DC	
0012F2E4	00110298	Arg1 = 00110298			
0012F2E8	00000002	Arg2 = 00000002			
0012F2EC	00000000	Arg3 = 00000000			
0012F2F0	00000000	Arg4 = 00000000			
0012F314	769EC4B7	Includes MFC42.6BAAA571	USER32.769EC4B4	0012F310	
0012F340	769EC5B7	? USER32.769EC494	USER32.769EC5B2	0012F33C	
0012F3B8	769E4EDE	? USER32.769EC504	USER32.769E4ED9	0012F3B4	

이번에는 Called from 쪽을 봐줘야한다. (왜냐하면 이거는 판단 로직이 아니라 그냥 실행을 안하게 만들거라서 지금 실행되고 있다는 뜻은 'Procedure' 보다는 결과를 실행하기 전의 함수를 보는거라서 'Called from'을 봐줘야한다.)

그럼 최근에 실행된 내부 함수인 VisualSi.00480C24가 보인다. 그걸 클릭해서 들어가면

CPU - main thread, module VisualSi			
00480C18	. 8D4C24 00	LEA ECX,DWORD PTR SS:[ESP]	
00480C1C	. C74424 68 00	MOV DWORD PTR SS:[ESP+68],0	
00480C24	. E8 01B80300	CALL <JMP.&MFC42.#2514>	
00480C29	. 8D4C24 00	LEA ECX,DWORD PTR SS:[ESP]	
00480C2D	. C74424 68 FF	MOV DWORD PTR SS:[ESP+68],-1	
00480C35	. E8 DEBA0300	CALL <JMP.&MFC42.#641>	
00480C3A	. 8B4C24 60	MOV ECX,DWORD PTR SS:[ESP+60]	
00480C3E	. 64:890D 0000	MOV DWORD PTR FS:[0],ECX	

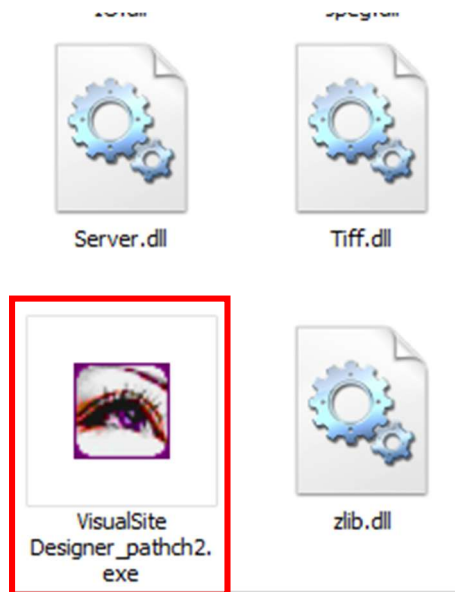
0x00480C24를 breakpoint를 걸어주고 다시 실행해보면 마지막 결제 유도 창이 발생하지 않는걸 볼 수 있다.

그럼 이 부분을 Binary > Fill with Nops로 바꿔준다.

00480C1C	. C74424 68 00	MOV DWORD PTR SS:[ESP+68],0	
00480C24	90	NOP	
00480C25	90	NOP	
00480C26	90	NOP	
00480C27	90	NOP	
00480C28	90	NOP	
00480C29	. 8D4C24 00	LEA ECX,DWORD PTR SS:[ESP]	
00480C2D	. C74424 68 FF	MOV DWORD PTR SS:[ESP+68],-1	
00480C35	. E8 DEBA0300	CALL <JMP.&MFC42.#641>	
00480C3A	. 8B4C24 60	MOV ECX,DWORD PTR SS:[ESP+60]	

이렇게 바뀌는 걸 볼 수 있다.

Copy to executable > Selection -> 덤프 창에서 Save File



실행해보면 마지막 결제 유도 창도 발생하지 않는 걸 볼 수 있다.

문제 해결!