

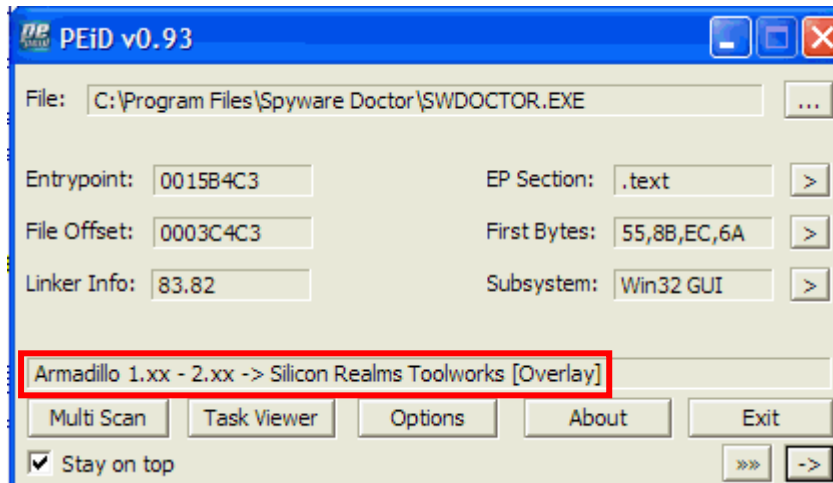
악성코드 분석 보고서

(sand-reversingwithlana-tutorials)

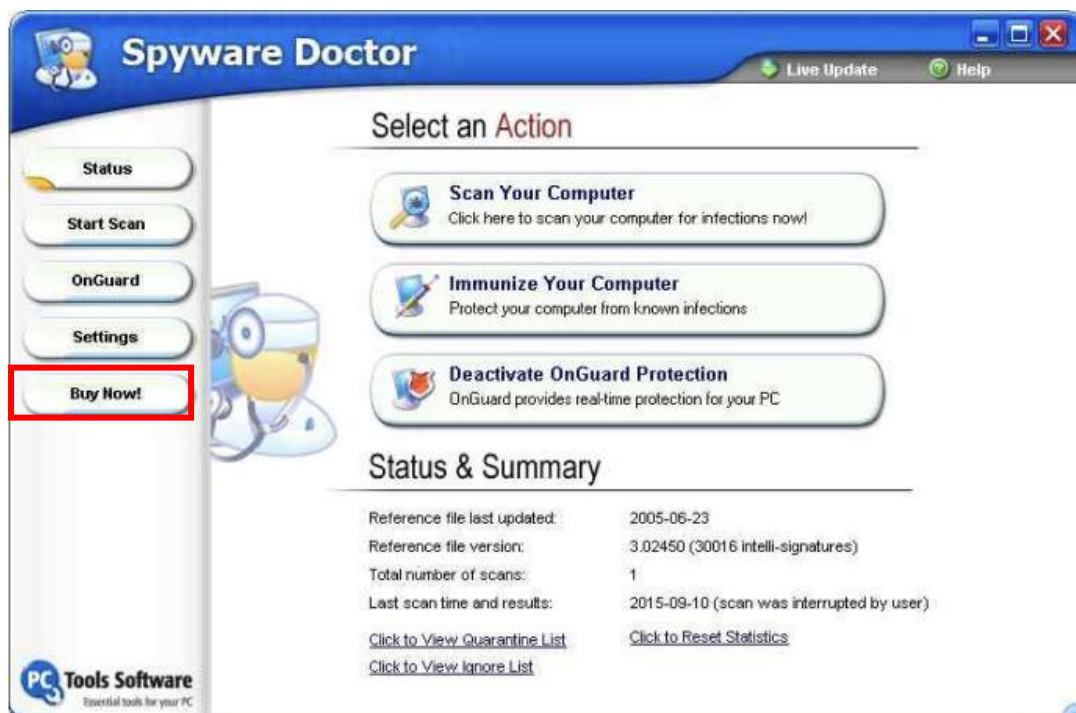
2025.10.13

24와 마찬가지로 실행이 되지 않아 lina영상 및 다른 사용자의 글을 참고하여 작성하였다.

1. SpywareDoctor.exe – Loader 사용



아르마딜로 패커를 사용한 걸 볼 수 있다.



Register Spyware Doctor

Enhanced program features require a licensed version of Spyware Doctor.

To upgrade to the licensed version, please click Purchase Online or click Continue to access the free version. Registered users, please enter your registration and license details below to activate the licensed version.

License Name

License Code

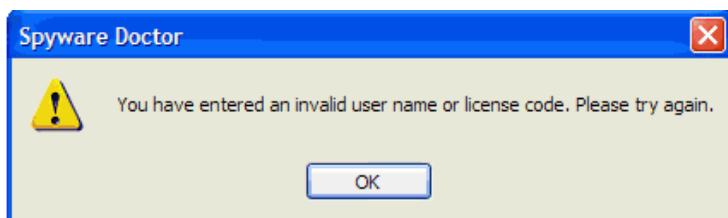
[Need help registering?](#)

Register Now

Purchase Online

Continue

레지스터 값을 입력하는 창이 생기는데 값을 입력해주고 'Register Now'를 눌러준다.



이렇게 입력한 값이 틀리다는 걸 알 수 있다.

K Call stack of main thread				
Address	Stack	Procedure / arguments	Called from	Frame
0012BBB8	77D19418	Includes ntdll.KiFastSystemCallRet	USER32.77D19416	0012BBEC
0012BBBC	77D2E2A2	USER32.WaitMessage	USER32.77D2E290	0012BBEC
0012BBF0	77D261C6	USER32.77D2E113	USER32.77D261C1	0012BBEC
0012BC18	77D3A92E	USER32.77D26110	USER32.77D3A929	0012BC14
0012BED8	77D3A294	USER32.SoftModalMessageBox	USER32.77D3A28F	0012BED4
0012C028	77D65FB6	USER32.77D3A11F	USER32.77D65FB6	0012C024
0012C080	77D66060	USER32.MessageBoxTimeoutW	USER32.77D6605B	0012C07C
0012C0B4	77D50577	? USER32.MessageBoxTimeoutA	USER32.77D50572	0012C0B0
0012C0D4	77D5052F	? USER32.MessageBoxExA	USER32.77D5052A	0012C0D0
0012C0D8	009F0298	hOwner = 009F0298 ('Spyware Doctor',class='TApplication')		
0012C0DC	01A09D68	Text = "You have entered an invalid user name or license code"		
0012C0E0	04E38384	Title = "Spyware Doctor"		
0012C0E4	00040030	Style = MB_OK MB_ICONEXCLAMATION MB_APPLMODAL:40000		
0012C0E8	00000000	LanguageID = 0 (LANG_NEUTRAL)		

올리디버그로 돌아간 후 콜스택을 보면 정보가 없는 걸 확인할 수 있다.

하지만 키 값을 입력한 후 'Register Now'를 누르면 bad boy창이 늦게 뜨는 걸 알 수 있다. 그래서 오래 걸리는 시간을 이용하여 일시정지를 해주고 콜 스택을 열어보도록 한다.

* 올리디버그 단축키

- Alt + F9 : 시스템/패커 루틴을 건너뛰고 사용자 코드 진입점으로 실행 (커서가 있는 부분에서 멈춤)

- Ctrl + F9 : 현재 함수가 RET 명령으로 끝날 때까지 실행

- F9 : 프로그램 전체 계속 실행 (다음 BP까지)

K Call stack of main thread				
Address	Stack	Procedure / arguments	Called from	Frame
001286A8	7C949FAC	ntdll.RtlFillMemoryUlong	ntdll.7C949FA7	001288D4
001288D8	7C96D6AA	? ntdll.7C91B2AC	ntdll.7C96D6A5	001288D4
0012895C	7C949D18	? ntdll.7C96D5FB	ntdll.7C949D13	00128958
0012888C	7C91B298	? ntdll.7C91B2AC	ntdll.7C91B293	00128888
00128DC0	77BFC3C9	? ntdll.RtlAllocateHeap	msvcrt.77BFC3C3	00128DBC
00128DC4	003D0000	hHeap = 003D0000		
00128DC8	40000060	Flags = HEAP_TAIL_CHECKING_ENABLED HEAP_FREE_CHECKING_ENABLED		
00128DC0	0000000A	HeapSize = A (10.)		
00128E00	77BFC3E7	? msvcrt.77BFC2E9		
00128E0C	77BF9CD4	msvcrt.77BFC3D4	msvcrt.77BFC3E2	00128DFC
00128E1C	00E61FEB	00E91370	msvcrt.77BF9CCF	00128E08
00128E2C	00E621AF	00E61FD4	00E61FE6	00128E18
00128E3C	00E6296C	00E6218E	00E621AA	00128E54
00128E58	00E62F3E	00E62874	00E62967	00128E54
00128E78	00E62E37	00E62F12	00E62F39	00128E54
00128E08	00E6342F	00E628BF	00E62E32	00128E74
00128F00	00E633CC	00E633F6	00E6342A	00128ED4
00128F34	00E63586	00E632F5	00E633C7	00128EFC
00128F80	00E771C4	00E63465	00E63581	00128F30
00129324	00E79D45	00E76D27	00E771BF	00128F7C
001296A4	00E7969F	00E7998C	00E79D40	00129320
00129CD8	00E791C8	00E791EF	00E7969A	001296A0
00129F1C	00E7B497	00E7914A	00E791C6	00129CD4
0012BF48	00E7B3E1	00E7B44F	00E7B492	00129F18
0012BF5C	00E8B8B8	00E7B3BC	00E7B3DC	0012BF44
0012C170	00E8B8B8	00E8B8B8	00E8B8B6	0012C174
0012C170	00E8B8B8	00E8B8B8	00E8B8B8	0012C174
0012C18C	00465F1D	swdoctor.00464AF0	swdoctor.00465F18	0012C1D0
0012C190	04E000F0	Arg1 = 04E000F0 ASCII "lena151"		
0012C194	04E38384	Arg2 = 04E38384 ASCII "55555555555555555555555555555555"		

이렇게 입력한 값이 swdocotr.00464AF0에 저장된 걸 볼 수 있고 0x00465F18에서 해당 함수가 불러진 걸 볼 수 있다. 0x00465F18 위치로 들어가본다.

CPU - main thread, module swdoctor				
00465F18	E8 D3EBFFFF	CALL swdoctor.00464AF0		
00465F19	84C0	TEST AL,AL		
00465F1F	74 3B	JE SHORT swdoctor.00465F5C		
00465F21	8D55 EC	LEA EDX,DWORD PTR SS:[EBP-14]		
00465F24	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]		
00465F27	8B80 10030000	MOV EAX,DWORD PTR DS:[EAX+310]		
00465F2D	E8 4EDAF9FF	CALL swdoctor.00403980		JMP to vc170.Controls::TControl::GetText
00465F32	8B45 EC	MOV EAX,DWORD PTR SS:[EBP-14]		
00465F35	E8 56B3F9FF	CALL swdoctor.00401290		JMP to rtl70.System::LStrToPChar
00465F3A	50	PUSH EAX		
00465F3B	8D55 E8	LEA EDX,DWORD PTR SS:[EBP-18]		
00465F3E	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]		
00465F41	8B80 0C030000	MOV EAX,DWORD PTR DS:[EAX+30C]		
00465F47	E8 34DAF9FF	CALL swdoctor.00403980		JMP to vc170.Controls::TControl::GetText
00465F4C	8B45 E8	MOV EAX,DWORD PTR SS:[EBP-18]		
00465F4F	E8 3CB3F9FF	CALL swdoctor.00401290		JMP to rtl70.System::LStrToPChar
00465F54	50	PUSH EAX		
00465F55	E8 9EEBFFFF	CALL swdoctor.00464AF0		
00465F5A	E8 05	JMP SHORT swdoctor.00465F61		
00465F5C	E8 AFEBFFFF	CALL swdoctor.00464B10		

※ 리버싱(역공학)은 약간의 '시행착오(trial and error)'가 섞여 있다. 하지만 리버싱의 진짜 묘미는 '추측'을 가능한 한 줄이고 논리적인 관찰로 추측해야한다. 조금만 살펴보면 (지금 처럼 압축 해제된 소프트웨어에서도 볼 수 있듯이), 이 부분이 등록 루틴(registration routine) 이라는 걸 꽤 높은 확신으로 알 수 있다.

00465EEA	8B80	CALL swdoctor.00403980	JMP to vc170.Controls::TControl::GetText
00465EF0	E8 8BD9F9FF	CALL swdoctor.00403980	ntdll.7C90EE18
00465EF5	8B45 F4	MOV EAX,DWORD PTR SS:[EBP-C]	JMP to rtl70.System::LStrToPChar
00465EF8	E8 93B3F9FF	CALL swdoctor.00401290	
00465EFD	50	PUSH EAX	
00465EFE	8D55 F0	LEA EDX,DWORD PTR SS:[EBP-10]	
00465F01	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
00465F04	8B80 0C030000	MOV EAX,DWORD PTR DS:[EAX+30C]	
00465F0A	E8 71DAF9FF	CALL swdoctor.00403980	JMP to vc170.Controls::TControl::GetText
00465F0F	8B45 F0	MOV EAX,DWORD PTR SS:[EBP-10]	
00465F12	E8 79B3F9FF	CALL swdoctor.00401290	JMP to rtl70.System::LStrToPChar
00465F17	50	PUSH EAX	
00465F18	E8 D3EBFFFF	CALL swdoctor.00464AF0	
00465F1D	84C0	TEST AL,AL	
00465F1F	74 3B	JE SHORT swdoctor.00465F5C	
00465F21	8D55 EC	LEA EDX,DWORD PTR SS:[EBP-14]	
00465F24	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
00465F27	8B80 10030000	MOV EAX,DWORD PTR DS:[EAX+310]	
00465F2D	E8 4EDAF9FF	CALL swdoctor.00403980	JMP to vc170.Controls::TControl::GetText
00465F32	8B45 EC	MOV EAX,DWORD PTR SS:[EBP-14]	
00465F35	E8 56B3F9FF	CALL swdoctor.00401290	JMP to rtl70.System::LStrToPChar
00465F3A	50	PUSH EAX	
00465F3B	8D55 E8	LEA EDX,DWORD PTR SS:[EBP-18]	
00465F3E	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
00465F41	8B80 0C030000	MOV EAX,DWORD PTR DS:[EAX+30C]	
00465F47	E8 34DAF9FF	CALL swdoctor.00403980	JMP to vc170.Controls::TControl::GetText

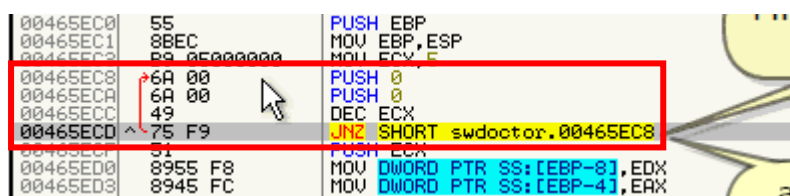
스크롤을 올리면 빨간색의 첫 번째 GetText에서는 시리얼을 읽는 부분, 두 번째 GetText는 이름을 읽는 부분이라는 걸 추정해볼 수 있고 그 후 값을 받아서 0x00464AF0으로 넘어가는 걸 볼 수 있고 그 후 JE 분기문이 실행되는 걸 볼 수 있다.

그리고 파란색의 첫 번째와 두 번째도 시리얼과 이름을 읽는 부분이라는 걸 추정할 수 있다. 하지만 JE가 실행된다면 이 부분은 실행되지 않을 것이다.

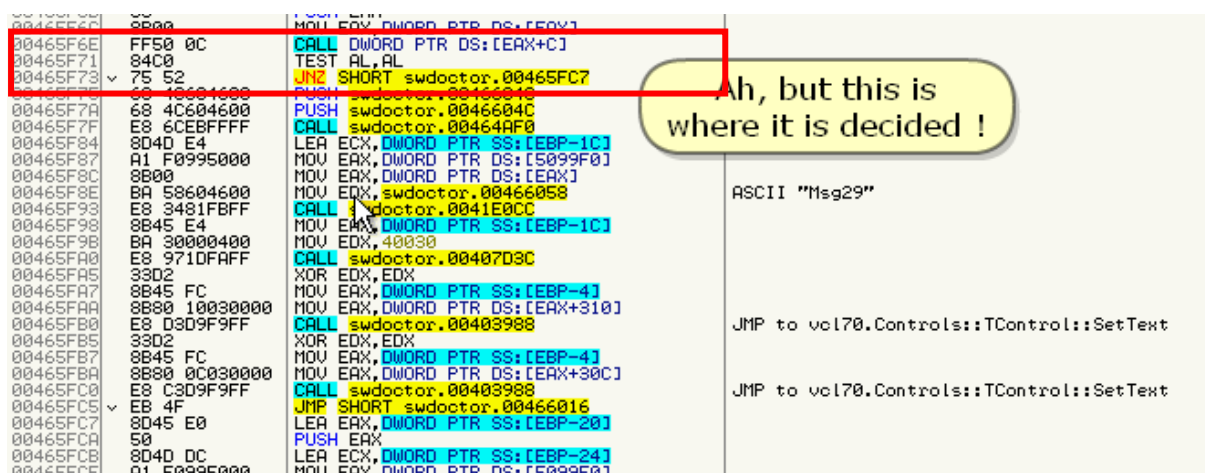
- GetText의 역할

GetText는 거의 항상 텍스트 속성을 읽어오는 메서드.

즉, 프로그램의 UI 요소(텍스트 박스, 레이블, 버튼 등)에 표시된 문자열을 가져오는 함수.



스크롤을 올리면 해당 부분을 볼 수 있는데 단지 스택을 준비하는(pushing 0) 동작을 하는 부분이다.



콜 스택에서 0x00465F18 기준으로 밑으로 내려가보면 또 하나의 분기문이 있는 걸 볼 수 있다. 해당 분기문이 점프를 하면 SetText가 실행 안되는 걸 볼 수 있다.

해당 부분은 실패 메시지를 보여주는 함수이므로 점프를 해야 성공 메시지를 띄울 수 있다는 것을 추정할 수 있다.

해당 분기문에서 메시지의 방향성을 선택하니까 그 위의 call 함수인 0x00465F6E는 시리얼과/이름을 비교해서 서로 맞으면 1, 다르면 0을 나오게 하는 걸 알 수 있다.

리턴 값이 1일 경우 분기되는 "465FC7" 주소를 등록 성공이라고 추정하였으니 실제로

맞는지 안 맞는지 테스트 해봐야 한다.

- SetText의 역할

UI 컨트롤의 "Text" 속성 값을 바꾸는 함수.

00465EC0	8BEC	PUSH EBP
00465EC1	B9 05000000	MOV EBP,ESP
00465EC3	6A 00	MOV ECX,5
00465EC8	6A 00	PUSH 0
00465ECA	49	PUSH 0
00465ECC	75 F9	DEC ECX
00465ECD	51	JNZ SHORT swdoctor.00465EC8
00465ECF	8955 F8	PUSH ECX
00465ED0	8945 FC	MOV DWORD PTR SS:[EBP-8],EDX
00465ED3	33C0	MOV DWORD PTR SS:[EBP-4],EAX
00465ED6	55	XOR EAX,EAX
00465ED8	55	PUSH EBP

추정한 것들이 맞는지 확인하기 위해서 처음부터 실행해봐야한다. 0x00465EC0 부분에 HW BP를 건 후 실행해준다.

Register Spyware Doctor

Enhanced program features require a licensed version of Spyware Doctor.

To upgrade to the licensed version, please click Purchase Online or click Continue to access the free version. Registered users, please enter your registration and license details below to activate the licensed version.

License Name

License Code

[Need help registering?](#)

00465EBF	90	NOP	
00465EC0	55	PUSH EBP	
00465EC1	8BEC	MOV EBP,ESP	
00465EC3	B9 05000000	MOV ECX,5	
00465EC8	6A 00	PUSH 0	
00465ECA	6A 00	PUSH 0	
00465ECC	49	DEC ECX	
00465ECD	75 F9	JNZ SHORT swdoctor.00465EC8	vol170.Stdctrls::TButton::CNCommand
00465ECF	51	PUSH ECX	vol170.Stdctrls::TButton::CNCommand
00465ED0	8955 F8	MOV DWORD PTR SS:[EBP-8],EDX	
00465ED3	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX	
00465ED6	33C0	XOR EAX,EAX	

값들을 입력 후 'Register Now'를 눌러주면 BP 부분에서 멈추는 걸 볼 수 있다.

00465EE7	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
00465EEA	8B80 10030000	MOV EAX,DWORD PTR DS:[EAX+310]	
00465EF0	E8 8BDAF9FF	CALL swdoctor.00403980	JMP to vol170.Controls::TControl::GetText
00465EF5	8B45 F4	MOV EAX,DWORD PTR SS:[EBP-C]	
00465EF8	E8 93B3F9FF	CALL swdoctor.00401290	JMP to rt170.System::LStrToPChar
00465FF0	50	PUSH FAX	

Stack SS:[0012C1C4]=04E38384, (ASCII "66666666666666666666666666666666")
EAX=0000001A

아까 빨간색의 첫 번째 부분을 보면 시리얼 값을 읽어드리는 걸 볼 수 있다.

00465F04	8B80 0C030000	MOV EAX,DWORD PTR DS:[EAX+30C]	
00465F08	E8 71DAF9FF	CALL swdoctor.00403980	JMP to vc170.Controls::TControl::GetText
00465F0F	8B45 F0	MOV EAX,DWORD PTR SS:[EBP-10]	
00465F12	E8 79B3F9FF	CALL swdoctor.00401290	JMP to rtl70.System::LStrToPChar

Stack SS:[0012C1C0]=04E539EC, (ASCII "lena151")
EAX=00000007

아까 빨간색의 두 번째 부분을 보면 이름 값을 읽어드리는 걸 볼 수 있다.

00465F10	84C0	TEST AL,AL	
00465F1F	74 3B	JE SHORT swdoctor.00465F5C	
00465F21	8D55 EC	LEA EDX,DWORD PTR SS:[EBP-14]	
00465F24	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
00465F27	8B80 10030000	MOV EAX,DWORD PTR DS:[EAX+310]	
00465F2D	E8 4EDAF9FF	CALL swdoctor.00403980	JMP to vc170.Controls::TControl::GetText
00465F32	8B45 EC	MOV EAX,DWORD PTR SS:[EBP-14]	
00465F35	FA 56B3F9FF	CALL swdoctor.00401290	JMP to rtl70.System::LStrToPChar

Jump is taken
00465F5C=swdoctor.00465F5C

그 후 파란색 부분인데 이 부분은 점프하는 걸 볼 수 있다. 해당 부분도 GetText를 사용하는 걸 볼 수 있는데 이 부분은 값을 입력 안하면 발생하는 코드로 알 수 있다.

00465F6C	8B80	MOV EAX,DWORD PTR DS:[EAX]	
00465F6E	FF50 0C	CALL DWORD PTR DS:[EAX+C]	
00465F71	84C0	TEST AL,AL	
00465F73	75 52	JNZ SHORT swdoctor.00465FC7	
00465F75	68 4C604600	PUSH swdoctor.0046604C	
00465F7A	68 4C604600	PUSH swdoctor.0046604C	
00465F7F	E8 6CEBFFFF	CALL swdoctor.00464AF0	
00465F84	8D4D E4	LEA ECX,DWORD PTR SS:[EBP-1C]	
00465F87	A1 F0995000	MOV EAX,DWORD PTR DS:[5099F0]	
00465F8C	8B80	MOV EAX,DWORD PTR DS:[EAX]	
00465F8E	BA 56604600	MOV EDI,DWORD PTR DS:[swdoctor.00466058]	ASCII "Msg29"
00465F93	E8 341E0000	CALL swdoctor.0041E0CC	
00465F98	8B45	MOV EAX,DWORD PTR SS:[EBP-1C]	
00465F9B	BA 30000400	MOV EDI,DWORD PTR DS:[swdoctor.00407D3C]	
00465FA0	E8 971DAFFF	CALL swdoctor.00407D3C	rtl70.400765D4
00465FA5	33D2	XOR EDI,EDI	
00465FA7	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
00465FAA	8B80 10030000	MOV EAX,DWORD PTR DS:[EAX+310]	
00465FB0	E8 D3D9F9FF	CALL swdoctor.00403988	JMP to vc170.Controls::TControl::SetText
00465FB5	33D2	XOR EDI,EDI	rtl70.400765D4
00465FB7	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
00465FBA	8B80 0C030000	MOV EAX,DWORD PTR DS:[EAX+30C]	
00465FC0	E8 C3D9F9FF	CALL swdoctor.00403988	JMP to vc170.Controls::TControl::SetText
00465FC5	EB 4F	JMP SHORT swdoctor.00466016	

Is this indeed
a bad sign ?

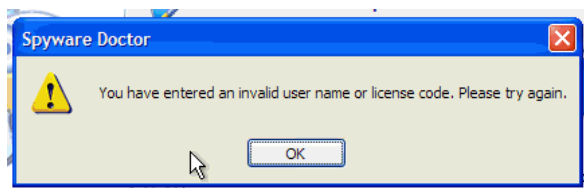
실행하다보면 SetText를 실행하는지 안하는지 판단하는 분기문에 도달하는 걸 볼 수 있다. 전에는 점프를 하면 성공 메시지, 점프를 하지 않으면 실패 메시지를 발생시킨다고 추정을 했는데 이번에 실행해보니까 점프가 실행안되는 걸 판단할 수 있다.

즉, 해당 부분은 추정하는게 맞았고 call함수가 판단하는 부분이라는 걸 알았다.

00465F8C	8B80	MOV EAX,DWORD PTR DS:[EAX]	
00465F8E	BA 56604600	MOV EDI,DWORD PTR DS:[swdoctor.00466058]	ASCII "Msg29"
00465F93	E8 341E0000	CALL swdoctor.0041E0CC	
00465F98	8B45 E4	MOV EAX,DWORD PTR SS:[EBP-1C]	
00465F9B	BA 30000400	MOV EDI,DWORD PTR DS:[swdoctor.00407D3C]	
00465FA0	E8 971DAFFF	CALL swdoctor.00407D3C	swdoctor.00466058
00465FA5	33D2	XOR EDI,EDI	
00465FA7	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
00465FAA	8B80 10030000	MOV EAX,DWORD PTR DS:[EAX+310]	
00465FB0	E8 D3D9F9FF	CALL swdoctor.00403988	JMP to vc170.Controls::TControl::SetText
00465FB5	33D2	XOR EDI,EDI	swdoctor.00466058
00465FB7	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
00465FBA	8B80 0C030000	MOV EAX,DWORD PTR DS:[EAX+30C]	
00465FC0	E8 C3D9F9FF	CALL swdoctor.00403988	JMP to vc170.Controls::TControl::SetText
00465FC5	EB 4F	JMP SHORT swdoctor.00466016	
00465FC7	8D45 E0	LEA EAX,DWORD PTR SS:[EBP-20]	
00465FCA	50	PUSH EAX	
00465FCB	8D4D DC	LEA ECX,DWORD PTR SS:[EBP-24]	
00465FCE	A1 F0995000	MOV EAX,DWORD PTR DS:[5099F0]	
00465FD3	8B80	MOV EAX,DWORD PTR DS:[EAX]	
00465FD5	BA 68604600	MOV EDI,DWORD PTR DS:[swdoctor.00466068]	ASCII "Msg28"
00465FDA	E8 ED80FBFF	CALL swdoctor.0041E0CC	
00465FDF	8B45 DC	MOV EAX,DWORD PTR SS:[EBP-24]	
00465FE2	50	PUSH EAX	
00465FE3	A1 8C985000	MOV EAX,DWORD PTR DS:[50988C]	
00465FE8	8B80	MOV EAX,DWORD PTR DS:[EAX]	
00465FEA	8945 D4	MOV DWORD PTR SS:[EBP-2C],EAX	
00465FED	C645 D8 0B	MOV BYTE PTR SS:[EBP-20],0B	
00465FF1	8D55 D4	LEA EDI,DWORD PTR SS:[EBP-2C]	
00465FF4	33C9	XOR ECX,ECX	
00465FF6	FA	REP FAX	0012C500

Preparing the
Badboy !

Stack SS:[0012C1B4]=01A09D68, (ASCII "You have entered an invalid user name or license code. Please try again.")
EAX=0012C1B8



실행을 해보면 발생시키는 걸 볼 수 있다. 다시 올리디버그로 돌아가본다.

00465FA5	3302	CALL swdoctor.0040703C	
00465FA7	8B45 FC	XOR EDX,EDX	
00465FA8	8B80 10030000	MOV EAX,DWORD PTR SS:[EBP-4]	
00465FAB	E8 D3D9F9FF	MOV EAX,DWORD PTR DS:[EAX+310]	
00465FAD	3302	CALL swdoctor.00403988	JMP to vol70.Controls::TControl::SetText
00465FAE	8B45 FC	XOR EDX,EDX	
00465FAC	8B80 0C030000	MOV EAX,DWORD PTR SS:[EBP-4]	
00465FAD	E8 C3D9F9FF	MOV EAX,DWORD PTR DS:[EAX+30C]	
00465FCE	EB 4F	CALL swdoctor.00403988	JMP to vol70.Controls::TControl::SetText
00465FCE	EB 4F	JMP SHORT swdoctor.00466016	

0x00465FA0에서 실패 메시지가 발생하는 걸 볼 수 있다. 아까 경고 창에서 'OK'버튼을 누르면 "SetText" 함수를 호출하여 입력 창을 비워서 사용자의 입력을 기다리는 걸 볼 수 있다.

00465F6B	50	PUSH EAX	
00465F6C	8B80	MOV EAX,DWORD PTR DS:[EAX]	
00465F6E	FF50 0C	CALL DWORD PTR DS:[EAX+C]	
00465F71	84C0	TEST AL,AL	
00465F73	75 52	JNZ SHORT swdoctor.00465FC7	
00465F75	68 4C604600	PUSH swdoctor.0046604C	
00465F7A	68 4C604600	PUSH swdoctor.0046604C	
00465F7F	E8 6CEBFFFF	CALL swdoctor.00464AF0	
00465F84	804D E4	LEA ECX,DWORD PTR SS:[EBP-1C]	
00465F87	A1 F0995000	MOV EAX,DWORD PTR DS:[5099F0]	
00465F8C	8B80	MOV EAX,DWORD PTR DS:[EAX]	
00465F8E	BA 58604600	MOV EDI,swdoctor.00466058	
00465F93	E8 3481FBFF	CALL swdoctor.0041E0CC	
00465F98	8B45 E4	MOV EAX,DWORD PTR SS:[EBP-1C]	
00465F9B	BA 30004000	MOV EDI,40030	

JNZ -> JMP로 바꾸면 되겠지만 이번에는 조건문(JNZ)에 사용되는 AL 레지스터 값을 세팅하는 0x00465F6E를 패치하는 "intermediate" 패치를 보여줄 것이다.

왜 이렇게 복잡한 패치를 진행하는지 궁금하다면, 만약 "JNZ" 조건문을 패치해 버리면 그냥 메인 루틴 부분만 패치되는 것이다. 따라서 다른 루틴에서 해당 함수를 호출해 버리면 무용 지물이 된다. 그러나 "[EAX+C]" 함수 내부에서 AL 레지스터 값을 패치해 버리면 다른 루틴에서 해당 함수를 호출해도 자동적으로 패치가 된다.

Register Spyware Doctor

Enhanced program features require a licensed version of Spyware Doctor.

To upgrade to the licensed version, please click Purchase Online or click Continue to access the free version. Registered users, please enter your registration and license details below to activate the licensed version.

License Name

License Code

[Need help registering?](#)

다시 값들을 입력 후 'Regiser Now'를 눌러준 후 0x00465F6E위치까지 와준다.

00465F6B	50	PUSH EAX	029B4
00465F6C	8B00	MOV EAX,DWORD PTR DS:[EAX]	swdoctor.00502995
00465F6E	FF50 0C	CALL DWORD PTR DS:[EAX]	swdoctor.0050296D
00465F71	84C0	TEST AL,AL	
00465F73	75 52	JNZ SHORT swdoctor.00465FC7	
00465F75	68 4C604600	PUSH swdoctor.0046604C	
00465F7A	68 4C604600	PUSH swdoctor.0046604C	
00465F7F	E8 6CEBFFFF	CALL swdoctor.00464AF0	
00465F84	8D4D E4	LEA ECX,DWORD PTR SS:[EBP-1C]	
00465F87	A1 F0995000	MOV EAX,DWORD PTR DS:[5099F0]	
00465F8C	8B00	MOV EAX,DWORD PTR DS:[EAX]	swdoctor.00502995
00465F8E	BA 58604600	MOV EDX,swdoctor.00466058	ASCII "Msg29"
00465F93	E8 3481FBFF	CALL swdoctor.0041E0CC	

해당 부분을 'F7'을 눌러 들어가준다.

0050296D	834424 04 F4	ADD DWORD PTR SS:[ESP+4],-0C	
00502972	E9 39060000	JMP swdoctor.00502FB0	
00502977	834424 04 F4	ADD DWORD PTR SS:[ESP+4],-0C	
0050297C	E9 4B0B0000	JMP swdoctor.005024CC	
00502981	834424 04 F4	ADD DWORD PTR SS:[ESP+4],-0C	
00502986	E9 D90B0000	JMP swdoctor.00502564	
0050298B	834424 04 F4	ADD DWORD PTR SS:[ESP+4],-0C	
00502990	E9 F3060000	JMP swdoctor.00502088	
00502995	834424 04 F4	ADD DWORD PTR SS:[ESP+4],-0C	
0050299A	E9 89EAEFFF	JMP swdoctor.00402028	
0050299F	834424 04 F4	ADD DWORD PTR SS:[ESP+4],-0C	
005029A4	E9 87EAEFFF	JMP swdoctor.00402030	
005029A9	834424 04 F4	ADD DWORD PTR SS:[ESP+4],-0C	
005029AE	E9 85EAEFFF	JMP swdoctor.00402038	

0x0050296D에 도달하고 'F8'을 이용해 하나 내려가 주면 JMP분기문이 실행되는 걸 볼 수 있을 것이다.

00502FB0	55	PUSH EBP	
00502FB1	8BEC	MOV EBP,ESP	
00502FB3	6A 00	PUSH 0	
00502FB5	6A 00	PUSH 0	
00502FB7	6A 00	PUSH 0	
00502FB9	33C0	XOR EAX,EAX	swdoctor.005029B4
00502FBB	55	PUSH EBP	
00502FBC	68 43305000	PUSH swdoctor.00433050	
00502FC1	64:FF30	PUSH 0	
00502FC4	64:8920	MOV DWORD PTR DS:[EAX],ESP	
00502FC7	8D55 F8	LEA EDX,DWORD PTR SS:[EBP-8]	
00502FCA	B8 54305000	MOV EAX,swdoctor.00503054	ASCII "USERNAME"
00502FCF	E8 BCFCEFFF	CALL swdoctor.00502C90	
00502FD4	837D F8 00	CMP DWORD PTR SS:[EBP-8],0	
00502FD8	74 22	JE SHORT swdoctor.00502FFC	
00502FDA	BA 68305000	MOV EDX,swdoctor.00503068	ASCII "DEFAULT"

0x0050296D에 HW BP를 설치하고 프로그램을 재실행해보니 해당 부분이 총 23번 호출되었다. 23번 호출되었다는 의미는 등록 상태를 검사하는 코드가 23번 사용되었다는 것이다.

00502FBB	55	PUSH EBP
00502FBC	68 43305000	PUSH swdoctor.00503043
00502FC1	64:FF30	PUSH DWORD PTR FS:[EAX]
00502FC4	64:8920	MOV DWORD PTR FS:[EAX],ESP
00502FC7	8D55 F8	LEA EDX,DWORD PTR SS:[EBP-8]
00502FCA	B8 54305000	MOV EAX,swdoctor.00503054
00502FCF	E8 BCFCFFFF	CALL swdoctor.00502C90
00502FD4	837D F8 00	CMP DWORD PTR SS:[EBP-8],0
00502FD8	74 22	JE SHORT swdoctor.00502FFC
00502FDA	BA 68305000	MOV EDX,swdoctor.00503068
00502FDF	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]
00502FE2	E8 59EFFFFF	CALL swdoctor.00401F40
00502FE7	85C0	TEST EAX,EAX
00502FE9	74 11	JE SHORT swdoctor.00502FFC
00502FEB	BA 78305000	MOV EDX,swdoctor.00503078
00502FE8	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]
00502FF3	E8 48EFFFFF	CALL swdoctor.00401F40
00502FF8	85C0	TEST EAX,EAX
00502FFA	75 06	JNZ SHORT swdoctor.00503002
00502FFC	C645 FF 00	MOV BYTE PTR SS:[EBP-14],0
00503000	EB 04	JMP SHORT swdoctor.00503006
00503002	C645 FF 01	MOV BYTE PTR SS:[EBP-13],1
00503006	807D FF 00	CMP BYTE PTR SS:[EBP-13],0
0050300A	74 13	JE SHORT swdoctor.0050301F
0050300C	8D55 F4	LEA EDX,DWORD PTR SS:[EBP-C]
0050300F	B8 80305000	MOV EAX,swdoctor.00503080
00503014	E8 77CFFFFF	CALL swdoctor.00502C90
00503019	837D F4 00	CMP DWORD PTR SS:[EBP-C],0
0050301D	75 04	JNZ SHORT swdoctor.00503023
0050301F	33C0	XOR EAX,EAX
00503021	EB 02	JMP SHORT swdoctor.00503025
00503023	B0 01	MOV AL,1
00503025	8B45 FF	MOV BYTE PTR SS:[EBP-1],0
00503028	33C0	XOR EAX,EAX
0050302A	5A	POP EDX
0050302B	59	POP ECX
0050302C	59	POP ECX
0050302D	61 0310	MOV DWORD PTR FS:[EAX],EDX
00503030	68 4A305000	PUSH swdoctor.00503040
00503035	8D45 F4	LEA EAX,DWORD PTR SS:[EBP-C]
00503038	BA 02000000	MOV EDX,
0050303D	E8 CEE1FFFF	CALL swdoctor.00401210
00503042	C3	RETN

'F8'을 이용해서 내려가다 보면 RETN이 있는 걸 볼 수 있다.

해당 빨간 박스를 보면 PUSH를 사용하고 RETN이 나오는 걸 보면 JMP를 의미하는 걸 알 수 있다. (PUSH + RETN == JMP)

※ 원래 PUSH -> CALL -> RETN != JMP 이다.

왜냐하면 CALL함수가 정상적으로 리턴하지 않으면 RETN에 도달하지 않기 때문이다.

1. PUSH addr + RET가 JMP addr인 이유

- PUSH <addr>: 스택에 <addr> 값을 넣음 (스택 포인터 SP가 감소하고 그 위치에 <addr> 저장).

- RET: 스택에서 값 하나를 팝해서(POP) 그 값을 리턴 주소로 사용해 점프함.

따라서 바로 이어서 PUSH target 다음에 RET이 실행되면, RET는 스택에서 target을 꺼내서 그 주소로 점프한다. 결과적으로 push target; ret는 jmp target과 동작이 동일하게 보인다.

2. 중간에 CALL이 있는 경우

- CALL 명령은 현재 위치(리턴 주소)를 스택에 푸시하고 호출한 함수로 점프한다.

그래서 만약 실행 흐름상 PUSH target 이후 사이에서 다른 PUSH/CALL/POP 등이 실행 된다면 스택의 최상단(RET가 pop할 값)이 바뀐다.

이러면 RET가 target이 아니라 그 사이에 쌓인 다른 값(예: CALL의 리턴 주소)을 팝해서 그쪽으로 점프하게 되므로 의도한 동작이 깨진다.

-> PUSH target — (CALL 어떤 함수) — RET를 **트램폴린(trampoline)** 형태라고 부른다.

- 트램폴린(trampoline)

트램폴린(trampoline) 은 어셈블리나 시스템 프로그래밍, 함수 호출 구조에서 “제어 흐름을 다른 코드로 ‘튀겨’ 보내는 중간 다리 역할”을 하는 짧은 점프용 코드 조각 이다.

00502FB8	55	PUSH EBP	
00502FBC	68 43305000	PUSH swdoctor.00503043	
00502FC1	64:FF30	PUSH DWORD PTR FS:[EAX]	
00502FC4	64:8920	MOV DWORD PTR FS:[EAX],ESP	
00502FC7	8055 F8	LEA EDX,DWORD PTR SS:[EBP-8]	
00502FCA	B8 54305000	MOV EAX,swdoctor.00503054	ASCII "USERNAME"
00502FCF	E8 BCFCFFFF	CALL swdoctor.00502C90	
00502FD3	74 22	JE SHORT swdoctor.00502FFC	
00502FDA	BA 68305000	MOV EDX,swdoctor.00503068	ASCII "DEFAULT"
00502FDF	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]	
00502FE2	E8 59EFEFFF	CALL swdoctor.00401F40	JMP to rtl70.Sysutils::CompareText
00502FE7	85C0	TEST EAX,EAX	
00502FE9	74 11	JE SHORT swdoctor.00502FFC	
00502FEB	BA 78305000	MOV EDX,swdoctor.00503078	ASCII "MISSING"
00502FF0	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]	
00502FF3	E8 4CFEFFFF	CALL swdoctor.00401F40	JMP to rtl70.Sysutils::CompareText
00502FF8	85C0	TEST EAX,EAX	

JE분기문에서 점프 안하는 걸 볼 수 있다. 근데 해당 부분은 CompareText가 있는 부분 이다. 그럼 이 부분에서 결정되는 걸 볼 수 있다.

00502FC7	8055 F8	LEA EDX,DWORD PTR SS:[EBP-8]		
00502FCA	B8 54305000	MOV EAX,swdoctor.00503054	ASCII "USERNAME"	
00502FCF	E8 BCFCFFFF	CALL swdoctor.00502C90		
00502FD3	74 22	JE SHORT swdoctor.00502FFC		
00502FDA	BA 68305000	MOV EDX,swdoctor.00503068	ASCII "DEFAULT"	
00502FDF	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]		
00502FE2	E8 59EFEFFF	CALL swdoctor.00401F40		
00502FE7	85C0	TEST EAX,EAX		
00502FE9	74 11	JE SHORT swdoctor.00502FFC		
00502FEB	BA 78305000	MOV EDX,swdoctor.00503078	ASCII "MISSING"	
00502FF0	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]		
00502FF3	E8 4CFEFFFF	CALL swdoctor.00401F40	JMP to rtl70	
00502FF8	85C0	TEST EAX,EAX		
00502FFA	75 06	JNZ SHORT swdoctor.00503002		
00502FFD	C645 FF 00	MOV BYTE PTR SS:[EBP-11],0		
00503000	EB 84	JMP SHORT swdoctor.00503006		
00503003	C645 FF 01	MOV BYTE PTR SS:[EBP-11],1		

Registers (FPU)
 EAX 00000000
 ECX 00000000
 EDI 00000007
 EBX 040F512C
 ESP 0012C174
 EBP 0012C18C
 ESI 01711684 vol170.StdCtrls::TButton
 EDI 0012C34C
 EIP 00502FE7 swdoctor.00502FE7
 C 0 ES 0023 32bit 0(FFFFFFFF)
 P 1 CS 001B 32bit 0(FFFFFFFF)
 A 0 SS 0023 32bit 0(FFFFFFFF)
 Z 1 DS 0023 32bit 0(FFFFFFFF)
 S 0 FS 003B 32bit 7FDD0000(FFF)
 T 0 GS 0000 NULL

But after the call,
 EAX = 0

EAX == 0이 나오는 걸 볼 수 있다.

00502FE7	85C0	TEST EAX, EAX	
00502FEB	74 11	JE SHORT swdoctor.00502FFC	ASCII "MISSING"
00502FF0	BA 78305000	MOV EDX, swdoctor.00503078	
00502FF3	8B45 F8	MOV EAX, DWORD PTR SS:[EBP-8]	
00502FF8	E8 48EFFFFF	CALL swdoctor.00401F40	JMP to rtl70.Sysutils::CompareText
00502FF8	85C0	TEST EAX, EAX	
00502FFA	75 06	JNZ SHORT swdoctor.00503002	
00502FFC	C645 FF 00	MOV BYTE PTR SS:[EBP-1], 0	
00503000	EB 04	JMP SHORT swdoctor.00503006	
00503002	C645 FF 01	MOV BYTE PTR SS:[EBP-1], 1	
00503006	807D FF 00	CMF BYTE PTR SS:[EBP-1], 0	
0050300A	74 13	JE SHORT swdoctor.0050301F	
0050300C	8D55 F4	LEA EDX, DWORD PTR SS:[EBP-C]	
0050300F	B8 80305000	MOV EAX, swdoctor.00503080	
00503014	E8 77CFFFFF	CALL swdoctor.00502C90	
00503019	837D F4 00	CMF DWORD PTR SS:[EBP-C], 0	
0050301D	75 04	JNZ SHORT swdoctor.00503023	
00503021	EB 02	JMP SHORT swdoctor.0050302F	
00503023	B0 01	MOV AL, 1	Aha, AL is set to 1 here = good news
00503025	8845 FF	MOV BYTE PTR SS:[EBP-1], AL	
00503028	33C0	XOR EAX, EAX	
0050302A	5A	POP EDX	
0050302B	59	POP ECX	
0050302C	59	POP ECX	
0050302D	64:8910	MOV DWORD PTR FS:[EAX], EDX	
00503030	68 4A305000	PUSH swdoctor.0050304A	
00503035	8D45 F4	LEA EAX, DWORD PTR SS:[EBP-C]	
00503038	B8 02000000	MOV EDX, 2	
0050303D	E8 CEE1EFFF	CALL swdoctor.00401210	JMP to rtl70.System::LStrArrayClr
00503042	C3	RETN	

첫 번째 빨간 박스를 보면 두 번째 CompareText가 실행안되는 걸 볼 수 있고 밑에 코드를 좀 더 보면 MOV AL, 1을 볼 수 있다. 그럼 앞에는 코드들은 다 상관없는 걸 볼 수 있다.

00503014	E8 77CFFFFF	CALL swdoctor.00502C90	
00503019	837D F4 00	CMF DWORD PTR SS:[EBP-C], 0	
0050301D	75 04	JNZ SHORT swdoctor.00503023	
00503021	90	NOP	
00503022	90	NOP	
00503023	B0 01	MOV AL, 1	
00503025	8845 FF	MOV BYTE PTR SS:[EBP-1], AL	
00503028	33C0	XOR EAX, EAX	

그럼 여기서 JMP분기문이 실행이 안되어야하므로 NOP으로 바꿔주면 된다.

0050301F	33C0	XOR EAX, EAX	
00503021	90	NOP	
00503022	90	NOP	
00503023	B0 01	MOV AL, 1	
00503025	8845 FF	MOV BYTE PTR SS:[EBP-1], AL	
00503028	5A	POP EDX	0012C18C
0050302B	59	POP ECX	0012C18C
0050302C	59	POP ECX	0012C18C
0050302D	64:8910	MOV DWORD PTR FS:[EAX], EDX	
00503030	68 4A305000	PUSH swdoctor.0050304A	
00503035	8D45 F4	LEA EAX, DWORD PTR SS:[EBP-C]	
00503038	B8 02000000	MOV EDX, 2	
0050303D	E8 CEE1EFFF	CALL swdoctor.00401210	JMP to rtl70.System::LStrArrayClr
00503042	C3	RETN	JMP to rtl70.System::HandleFinally
00503043	E9 70E1EFFF	JMP swdoctor.004011B8	
00503044	8A45 FF	MOV AL, BYTE PTR SS:[EBP-1]	
0050304D	8845 FF	MOV ECX, EBP	
0050304F	5D	POP EBP	0012C18C
00503050	C2 0400	RETN 4	
00503053	0055 53	ADD BYTE PTR SS:[EBP+53], DL	

AL이 [EBP-1]에 저장되는 걸 볼 수 있고 RETN이 끝나고 다시 AL이 [EBP-1]에 저장되어 있는 값을 가져 오는 걸 볼 수 있다.

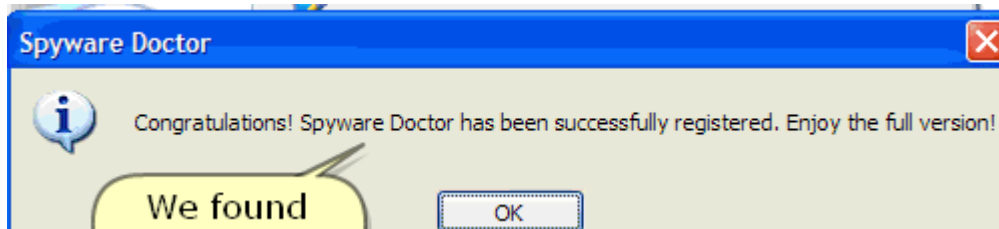
00465F71	84C0	TEST AL, AL	
00465F73	75 52	JNZ SHORT swdoctor.00465FC7	
00465F75	68 4C604600	PUSH swdoctor.0046604C	
00465F7A	68 4C604600	PUSH swdoctor.0046604C	
00465F7F	E8 6CEBFFFF	CALL swdoctor.00464AF0	
00465F84	8D4D E4	LEA ECX, DWORD PTR SS:[EBP-10]	
00465F87	A1 F0995000	MOV EAX, DWORD PTR DS:[509950F0]	

RETN 4까지 끝나야 내부 함수에서 나오는 걸 볼 수 있다.

00465F8A	BA 00007000	MOV EDI, swdoctor.00466000	
00465F8D	8B45 DC	MOV EAX, DWORD PTR SS:[EBP-24]	
00465F92	50	PUSH EAX	
00465F93	A1 8C985000	MOV EAX, DWORD PTR DS:[5098508C]	
00465F96	0000	MOV EAX, DWORD PTR DS:[EAX]	

```
Stack SS:[0012C1AC]=01A0A778, (ASCII "Congratulations! %s has been successfully registered. Enjoy the full version!")  
EAX=0012C184
```

'F8'을 이용해서 실행해보면 0x00465FDF에서 성공 메시지가 나오는 걸 볼 수 있다.



해당 창이 나오고 'F9'를 눌러서 실행해준다.



"Buy Now"가 없어진 창이 뜨는 걸 볼 수 있다.

이제 해야 할 일은 "0x00503021" 주소 명령어를 패치(opcode 패치, "EB 02 => 90 90" == "JMP" => "NOP") 하는 로더를 만드는 것이다. 이전 레벨과 마찬가지로 "RISC Process Pathcer"를 이용할 것이다.



위와 같이 스크립트 파일을 만들고 "R!SC Process Pathcer"를 이용하여 로더를 제작하면 이제 프로그램이 실행 될 때마다 임의의 시리얼 값으로 등록이 될 것이다.