

XoftSpy 4.13



Registration

In order to remove the selected objects you need to Register your XoftSpy now.
Please enter Registration information provided.

If you do not have Registration information, click on "Get Registration Code" button or the link below

Get Registration Code

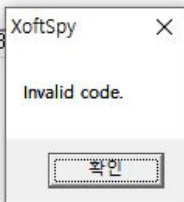
<http://www.paretologic.com/xoftspy/register.asp>

Enter Username:

Enter Registration Code:

 -

OK



xoftspy라는 프로그램은 윈도우용
스파이웨어 방지 프로그램으로 USB
드라이브에서 실행되는 일종의 백신
프로그램이다.

이번 프로그램도 등록 우회 하는게
목적이다.

이와 같이 이름을 입력하고, 등록
코드를 입력하면 정식 버전을 이용할
수 있다.

R Find: GETWINDOWT

Address	Disassembly	Destination
0046283C	CALL DWORD PTR DS:[<&USER32.GetWindowLongA	USER32.GetWindowLongA
00463080	CALL DWORD PTR DS:[<&USER32.GetWindowLongA	USER32.GetWindowLongA
00401E05	CALL DWORD PTR DS:[<&USER32.GetWindowRect	USER32.GetWindowRect
0040983F	CALL DWORD PTR DS:[<&USER32.GetWindowRect	USER32.GetWindowRect
0040A236	CALL DWORD PTR DS:[<&USER32.GetWindowRect	USER32.GetWindowRect
00419F82	CALL DWORD PTR DS:[<&USER32.GetWindowRect	USER32.GetWindowRect
00419FE7	CALL DWORD PTR DS:[<&USER32.GetWindowRect	USER32.GetWindowRect
00421967	CALL DWORD PTR DS:[<&USER32.GetWindowRect	USER32.GetWindowRect
00421A63	CALL DWORD PTR DS:[<&USER32.GetWindowRect	USER32.GetWindowRect
0042C7E4	CALL DWORD PTR DS:[<&USER32.GetWindowRect	USER32.GetWindowRect
0042F816	CALL DWORD PTR DS:[<&USER32.GetWindowRect	USER32.GetWindowRect
004323E4	CALL EBX	USER32.GetWindowRect
00447057	CALL DWORD PTR DS:[<&USER32.GetWindowRect	USER32.GetWindowRect
0045AA1A	CALL DWORD PTR DS:[<&USER32.GetWindowRect	USER32.GetWindowRect
0045AA56	CALL DWORD PTR DS:[<&USER32.GetWindowRect	USER32.GetWindowRect
0045C73A	CALL DWORD PTR DS:[<&USER32.GetWindowRect	USER32.GetWindowRect
0045CF00	CALL EBX	USER32.GetWindowRect
00461DFD	CALL DWORD PTR DS:[<&USER32.GetWindowRect	USER32.GetWindowRect
0040709A	CALL DWORD PTR DS:[<&KERNEL32.GetWindowsDirectoryA	KERNEL32.GetWindowsDirectoryA
004388B6	CALL DWORD PTR DS:[<&KERNEL32.GetWindowsDirectoryA	KERNEL32.GetWindowsDirectoryA
0045B75D	CALL DWORD PTR DS:[<&USER32.GetWindowTextA	USER32.GetWindowTextA
00461E51	CALL DWORD PTR DS:[<&USER32.GetWindowTextA	USER32.GetWindowTextA
00462596	CALL DWORD PTR DS:[<&USER32.GetWindowTextA	USER32.GetWindowTextA
0045B745	CALL DWORD PTR DS:[<&USER32.GetWindowTextLengthA	USER32.GetWindowTextLengthA
00462596	CALL DWORD PTR DS:[<&USER32.GetWindowTextLengthA	USER32.GetWindowTextLengthA
0045B1A2	CALL DWORD PTR DS:[<&KERNEL32.GlobalAddAtomA	KERNEL32.GlobalAddAtomA
00464836	CALL EBX	KERNEL32.GlobalAddAtomA
00464855	CALL EBX	KERNEL32.GlobalAddAtomA

이번 리버싱의 목적은 all intermodules calls를 이용해 전체문자열을 검색해 오류를 패치하는 것이 아닌 문자열이 없을 시 해당 프로그램에서 호출하는 API를 이용해 패치하는 것이 목적이다.

ollydbg all intermodules calls에서 GetWindowTextA를 검색해 해당 함수를 호출하는 모든 부분에 bp를 건다.

00462595	. 56	PUSH ESI	
00462596	. FF15 F8054700	CALL DWORD PTR DS:[<&USER32.GetWindowTextLengthF	hWnd
0046259C	. 8048 01	LEA ECX,DWORD PTR DS:[EAX+1]	GetWindowTextLengthF
0046259F	. 51	PUSH ECX	
004625A0	. 8B4D 10	MOV ECX,DWORD PTR SS:[EBP+10]	
004625A3	. 50	PUSH EAX	
004625A4	. E8 52C0FFFF	CALL XoftSpy_.0045E5FB	
004625A9	. 50	PUSH EAX	
004625AA	. 56	PUSH ESI	Buffer
004625AB	. FF15 F0034700	CALL DWORD PTR DS:[<&USER32.GetWindowTextA	hWnd
004625B1	. 8B4D 10	MOV ECX,DWORD PTR SS:[EBP+10]	GetWindowTextA
004625B4	. 6A FF	PUSH -1	
004625B6	. E8 18C0FFFF	CALL XoftSpy_.0045E5D3	
004625BB	. EB 0B	JMP SHORT XoftSpy_.004625C8	
004625BD	> 8B45 10	MOV EAX,DWORD PTR SS:[EBP+10]	

코드를 입력하고 **OK** 버튼을 누르면 해당 영역에서 멈추게 된다.

bp 아래 하나의 함수가 보인다. 실행해보니 해당 함수는 사용자가 입력한 코드 이름을 읽어오는 코드다.

등록 코드 앞에 뒤에 그리고 이름까지 총 세번 반복하면서 읽어온다.

004173D2	. 8BCD	MOV ECX,EBP	
004173D4	. E8 870E0400	CALL XoftSpy_.00458260	
004173D9	. 68 48FA4800	PUSH XoftSpy_.0048FA48	Arg2 = 0048FA48
004173DE	. 68 C0594800	PUSH XoftSpy_.004859C0	Arg1 = 004859C0
004173E3	. 8BCF	MOV ECX,EDI	
004173E5	. E8 C5060400	CALL XoftSpy_.00457AAF	XoftSpy_.00457AAF
004173EA	. 8B45 00	MOV EAX,DWORD PTR SS:[EBP]	
004173ED	. 8B40 F8	MOV EAX,DWORD PTR DS:[EAX-8]	
004173F0	. 85C0	TEST EAX,EAX	
004173F2	~0F84 BA010000	JE XoftSpy_.004175B2	
004173F8	. 8B0F	MOV ECX,DWORD PTR DS:[EDI]	
004173FA	. 8B41 F8	MOV EAX,DWORD PTR DS:[ECX-8]	
004173FD	. 85C0	TEST EAX,EAX	
004173FF	~0F84 AD010000	JE XoftSpy_.004175B2	
00417405	. 55	PUSH EBP	
00417406	. 8D4C24 18	LEA ECX,DWORD PTR SS:[ESP+18]	
0041740A	. E8 066B0400	CALL XoftSpy_.0045DF15	
0041740F	. 8D4C24 14	LEA ECX,DWORD PTR SS:[ESP+14]	

분석하다 보면 해당 분기문을 만나게 된다.

레지스터를 보니 내가 입력한 시리얼 값과 글자수가 적혀있었다.

그리고 분기하는 주소로 이동해보니 정확하게 입력하라는 오류가 있다. 그래서 해당 부분은 사용자가 필드를 모두 채우지 않았을 경우 보여지는 오류라고 추측할 수 있다.

Registers (FPU)	
EAX	00000008
ECX	028B8BF0 ASCII "124-1234"
EDX	028B8BF1 ASCII "24-1234"
EBX	0019F4CC
ESP	00199A80
EBP	0019F4D0
ESI	0019EDB0 ASCII "X G"
EDI	0019F4C8
EIP	004173FF XoftSpy_.004173FF
C 0	ES 002B 32bit 0(FFFFFFFF)

004175A1	. 5D	POP EBP	
004175A0	. 5B	POP EBX	
004175AE	. 83C4 1C	ADD ESP,1C	
004175B1	. C3	RETN	
004175B2	> 6A 00	PUSH 0	
004175B4	. 68 98524800	PUSH XoftSpy_.00485298	ASCII "Error"
004175B9	. 68 7C684800	PUSH XoftSpy_.0048687C	ASCII "Please fill all the details"
004175BE	. 8BCE	MOV ECX,ESI	
004175C0	. E8 524E0400	CALL XoftSpy_.0045C417	
004175C5	. 8B4C24 20	MOV ECX,DWORD PTR SS:[ESP+20]	
004175C9	. 5F	POP EDI	
004175CA	. 5E	POP ESI	

00417483	. 50	PUSH EAX	
00417484	. C64424 34 03	MOV BYTE PTR SS:[ESP+34],3	
00417489	. E8 876A0400	CALL XoftSpy_.0045DF15	
0041748E	. 8BCE	MOV ECX,ESI	
00417490	. C64424 30 01	MOV BYTE PTR SS:[ESP+30],1	
00417495	. E8 F6010000	CALL XoftSpy_.00417690	XoftSpy_.00417690
0041749A	. 84C0	TEST AL,AL	
0041749C	. 75 45	JNZ SHORT XoftSpy_.004174E3	
0041749E	. 6A 00	PUSH 0	
004174A0	. 68 04544800	PUSH XoftSpy_.00485404	ASCII "XoftSpy"
004174A5	. 68 C4684800	PUSH XoftSpy_.004868C4	ASCII "Invalid code."
004174AA	. 8BCE	MOV ECX,ESI	
004174AC	. E8 664F0400	CALL XoftSpy_.0045C417	
004174B1	. 70 40F04000	PUSH 0	

그리고 내려가다 보면 분기문이 또 있는데, 해당 분기문은 417690 함수의 리턴값이 0이 되면 분기가 일어나지 않아 잘못된 코드 오류를 반환한다.

00417457	. 8BCE	MOV ECX,ESI	
00417459	. C64424 30 01	MOV BYTE PTR SS:[ESP+30],1	
0041745E	. E8 2D020000	CALL XoftSpy_.00417690	XoftSpy_.00417690
00417463	. 84C0	TEST AL,AL	
00417465	. 75 7C	JNZ SHORT XoftSpy_.004174E3	
00417467	. 51	PUSH ECX	
00417468	. 8D5424 14	LEA EDX,DWORD PTR SS:[ESP+14]	
0041746C	. 8BCC	MOV ECX,ESP	
0041746E	. 896424 20	MOV DWORD PTR SS:[ESP+20],ESP	
00417472	. 52	PUSH EDX	
00417473	. E8 9D6A0400	CALL XoftSpy_.0045DF15	
00417478	. 51	PUSH ECX	
00417479	. 8D4424 1C	LEA EAX,DWORD PTR SS:[ESP+1C]	
0041747D	. 8BCC	MOV ECX,ESP	
0041747F	. 896424 20	MOV DWORD PTR SS:[ESP+20],ESP	
00417483	. 50	PUSH EAX	
00417484	. C64424 34 03	MOV BYTE PTR SS:[ESP+34],3	
00417489	. E8 876A0400	CALL XoftSpy_.0045DF15	
0041748E	. 8BCE	MOV ECX,ESI	
00417490	. C64424 30 01	MOV BYTE PTR SS:[ESP+30],1	
00417495	. E8 F6010000	CALL XoftSpy_.00417690	XoftSpy_.00417690
0041749A	. 84C0	TEST AL,AL	
0041749C	. 75 45	JNZ SHORT XoftSpy_.004174E3	
0041749F	. 6A 00	PUSH 0	

분기문 이전에 똑같은 함수를 두 번 호출해 AL값을 결정짓는걸 알 수 있다. 첫번째 함수에 bp를 걸고 분석해봐야 할 것 같다.

00417903	. 8D4C24 3C	LEA ECX,DWORD PTR SS:[ESP+3C]
00417907	. C74424 30 FFF	MOV DWORD PTR SS:[ESP+30],-1
0041790F	. E8 8C680400	CALL XoftSpy_.0045E1A0
00417914	. 8B4C24 28	MOV ECX,DWORD PTR SS:[ESP+28]
00417918	. 5F	POP EDI
00417919	. 5E	POP ESI
0041791A	. 32C0	XOR AL,AL
0041791C	. 5B	POP EBX
0041791D	. 64:8900 00000	MOV DWORD PTR FS:[0],ECX
00417924	. 83C4 28	ADD ESP,28
00417927	. C2 0800	RETN 8
0041792A	. 90	NOP

아까 **bp**를 걸었던 함수를 진입해 분석하다 보면 마지막에 **RETN** 명령어를 볼 수 있다. 그 전에 명령어를 살펴보면 **XOR AL,AL** 명령으로 **AL** 값을 **0**으로 만드는걸 볼 수 있다. 여태까지 분석한걸 바탕으로 설명하면 **0**을 반환하면 오류를 내뱉는다. 그렇기에 해당 부분은 호출되면 안된다.

00417894	. E8 07690400	CALL XoftSpy_.0045E1A0
00417899	. 5F	POP EDI
0041789A	. 5E	POP ESI
0041789B	. B0 01	MOV AL,1
0041789D	. 5B	POP EBX
0041789E	. 8B4C24 1C	MOV ECX,DWORD PTR SS:[ESP+1C]
004178A2	. 64:8900 00000	MOV DWORD PTR FS:[0],ECX
004178A9	. 83C4 28	ADD ESP,28
004178AC	. C2 0800	RETN 8
004178AF	> 8D4C24 10	LEA ECX,DWORD PTR SS:[ESP+10]
004178B3	. C64424 30 05	MOV BYTE PTR SS:[ESP+30],5
004178B8	. E8 E3680400	CALL XoftSpy_.0045E1A0
004178BD	. 8D4C24 14	LEA ECX,DWORD PTR SS:[ESP+14]

그래서 이전 명령어를 살펴봤는데, **4178AC** 부분 전에 보면 **AL**값을 설정하는 부분이 나온다. 이 부분에서 **AL**을 **1**로 설정하므로 해당 부분이 호출되게 해야된다. 어떤 부분에서 해당 부분을 호출하는지 살펴본다.

004176D6	. 51	PUSH ECX	Arg1 XoftSpy_.00448D4C
004176D7	. E8 70160300	CALL XoftSpy_.00448D4C	
004176DC	. 83C4 08	ADD ESP,8	
004176DF	. 85C0	TEST EAX,EAX	
004176E1	.v0F84 0E020000	JE XoftSpy_.004178F5	
004176E7	. 8B5424 3C	MOV EDX,DWORD PTR SS:[ESP+3C]	
004176EB	. 837A F8 15	CMP DWORD PTR DS:[EDX-8],15	
004176EF	.v0F8C 00020000	JL XoftSpy_.004178F5	
004176F5	. 8D4424 1C	LEA EAX,DWORD PTR SS:[ESP+1C]	
004176F9	. 6A 0B	PUSH 0B	
004176FB	. 50	PUSH EAX	
004176FC	. 8D4C24 44	LEA ECX,DWORD PTR SS:[ESP+44]	
00417700	. E8 93060400	CALL XoftSpy_.00457D98	

Address	Hex dump	ASCII
05A1F188	09 00 00 00 40 00 00 00
05A1F190	31 32 33 34 2D 31 32 33	1234
05A1F198	34 00 74 73 00 00 67 73	4.ts

프로그램을 살펴보니 어떤 부분도 **AL**, 1부분으로 분기하지 않았다. 그래서 살펴보니 해당 부분에서 **AL**, 0으로 점프를 시도하는걸 볼 수 있다. 분기하는 이유를 분석해보니 **CMP PTR DS:[EDX-8], 15**부분에서 비교하고 이 값보다 작을 경우 분기를 시도한다.

EDX-8부분을 살펴보면 9라는 값이 들어가 있는데, 이 값은 내가 입력한 코드의 길이를 나타내는것을 알 수 있다. 즉, 코드 길이는 15이상이어야 한다.

004176DC	. 83C4 08	MOV ESP,8
004176DF	. 85C0	TEST EAX,EAX
004176E1	✓ 0F84 0E020000	JE XoftSpy_.004178F5
004176E7	. 8B5424 3C	MOV EDX,DWORD PTR SS:[ESP+3C]
004176EB	. 837A F8 15	CMP DWORD PTR DS:[EDX-8],15
004176EF	✓ 0F8C 00020000	JL XoftSpy_.004178F5
004176F5	. 8D4424 1C	LEA EAX,DWORD PTR SS:[ESP+1C]
004176F9	. 6A 0B	PUSH 0B
004176FB	. 50	PUSH EAX
004176FC	. 8D4C24 44	LEA ECX,DWORD PTR SS:[ESP+44]
00417700	. E8 93060400	CALL XoftSpy_.00457D98
00417705	. 8B4C24 1C	MOV ECX,DWORD PTR SS:[ESP+1C]
00417709	. 33F6	XOR ESI,ESI

분기하는 부분의 S플래그를 조작해
분기하지 않도록 한다.

그리고 계속 분석해보면 두개의
반복문을 만나게되는데, 해당 부분은
등록 코드의 '-' 기준으로 앞뒤로 잘라
각각 EDX, ESI에 저장한다.

00417700	. C64424 30 02	MOV BYTE PTR SS:[ESP+30],2
00417712	. 897424 20	MOV DWORD PTR SS:[ESP+20],ESI
00417716	. 33C0	XOR EAX,EAX
00417718	> 0FB83C08	MOVSX EDI, BYTE PTR DS:[EAX+ECX]
0041771C	. 03D7	ADD EDX,EDI
0041771E	. 40	INC EAX
0041771F	. 83F8 0A	CMP EAX,0A
00417722	✓ 7E F4	JLE SHORT XoftSpy_.00417718
00417724	. 895424 24	MOV DWORD PTR SS:[ESP+24],EDX
00417728	. 8B5424 38	MOV EDX,DWORD PTR SS:[ESP+38]
0041772C	. 33C0	XOR EAX,EAX
0041772E	. 8B4A F8	MOV ECX,DWORD PTR DS:[EDX-8]
00417731	. 85C9	TEST ECX,ECX
00417733	✓ 7E 0F	JLE SHORT XoftSpy_.00417744
00417735	> 0FB83C10	MOVSX EDI, BYTE PTR DS:[EAX+EDX]
00417739	. 03F7	ADD ESI,EDI
0041773B	. 40	INC EAX
0041773C	. 3BC1	CMP EAX,ECX
0041773E	✓ 7C F5	JL SHORT XoftSpy_.00417735
00417740	. 897424 20	MOV DWORD PTR SS:[ESP+20],ESI
00417744	> 8B4C24 3C	MOV ECX,DWORD PTR SS:[ESP+3C]
00417748	. 55	PUSH EBP
00417749	. 8D5424 10	LEA EDX,DWORD PTR SS:[ESP+10]

004177F1	. 8B58 F8	MOV EBP,DWORD PTR DS:[EAX-8]
004177F4	. 3BF1	CMP ESI,ECX
004177F6	. 0FB50 01	MOVSX EDX,BYTE PTR DS:[EAX+1]
004177FA	. 0FBE4428 FF	MOVSX EAX,BYTE PTR DS:[EAX+EBP-
004177FE	. 5D	POP EBP
00417800	. ✓0F85 A9000000	JNZ XoftSpy_.004178AF
00417806	. 3BFA	CMP EDI,EDX
00417808	. ✓0F85 A1000000	JNZ XoftSpy_.004178AF
0041780E	. 3BD8	CMP EBX,EAX
00417810	. ✓0F85 99000000	JNZ XoftSpy_.004178AF
00417816	. 8B4424 20	MOV EAX,DWORD PTR SS:[ESP+20]
0041781A	. B9 0A000000	MOV ECX,0A
0041781F	. 99	CNO

또 내려가다보면 세개의 분기문이
나오고 세개 모두 다 같은 곳으로
분기한다.

해당 부분은 AL, 0으로 가는
주소이다. Z 플래그를 바꿔가며
우회해준다.

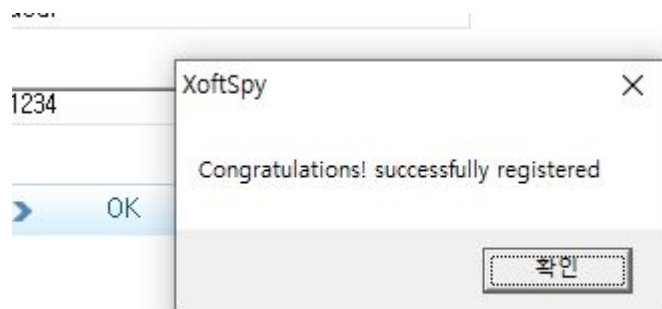
0041787D	. E8 20690400	CALL XoftSpy_.0045E1A0
0041787A	. 8D4C24 38	LEA ECX,DWORD PTR SS:[ESP+38]
0041787E	. C64424 30 00	MOV BYTE PTR SS:[ESP+30],0
00417883	. E8 18690400	CALL XoftSpy_.0045E1A0
00417888	. 8D4C24 3C	LEA ECX,DWORD PTR SS:[ESP+3C]
0041788C	. C74424 30 FFFF	MOV DWORD PTR SS:[ESP+30],-1
00417894	. E8 07690400	CALL XoftSpy_.0045E1A0
00417899	. 5F	POP EDI
0041789A	. 5E	POP ESI
0041789B	. B0 01	MOV AL,1
0041789D	. 5B	POP EBX
0041789E	. 8B4C24 1C	MOV ECX,DWORD PTR SS:[ESP+1C]
004178A2	. 64:890D 000000	MOV DWORD PTR FS:[0],ECX
004178A9	. 83C4 28	ADD ESP,28

플래그를 변경해가며
진행하다보면, AL, 1부분에
도달하게 된다.

00417452	. E8 BE000400	CALL XoftSpy_.00417453	
00417457	. 8BCE	MOV ECX,ESI	
00417459	. C64424 30 01	MOV BYTE PTR SS:[ESP+30],1	
0041745B	. E8 2D020000	CALL XoftSpy_.00417690	XoftSpy_.00417690
00417463	. 84C0	TEST AL,AL	
00417465	. 75 7C	JNZ SHORT XoftSpy_.004174E3	
00417467	. 51	PUSH ECX	
00417468	. 8D5424 14	LEA EDX,DWORD PTR SS:[ESP+14]	
0041746C	. 8BCC	MOV ECX,ESP	
0041746F	. 8B00	MOV ECX,DWORD PTR SS:[ESP+00]	

AL 부분을 조작했으니 아까 분기 못했던 JNZ 4174E3도 분기가 된다.

이렇게 성공 메시지를 볼 수 있다.



About

This XoftSpy license has not been registered

<http://www.paretologic.com/xoftspy>

<http://www.paretologic.com/support>

Copyright ?2004 ParetoLogic Inc.
All rights reserved.

[Terms of use](#)

등록에 성공하였음에도 불구하고, **About** 메뉴에서는 여전히 등록이 아직 안됐다는 메시지를 보여주고 있다. 이 부분도 패치해주기 위해 분석해본다.

이번에는 **API**가 아닌 문자열로 검색한다.

00401489	. 8B8E C0000000	MOV ECX,DWORD PTR DS:[ESI+C0]	
0040148F	. 83C4 08	ADD ESP,8	
00401492	. E8 093C0300	CALL XoftSpy_.004350A0	
00401497	. 84C0	TEST AL,AL	
00401499	✓74 12	JE SHORT XoftSpy_.004014AD	
0040149B	. 68 C4514800	PUSH XoftSpy_.004851C4	ASCII "This license of XoftSpy has been registered"
004014A0	. 8D4C24 08	LEA ECX,DWORD PTR SS:[ESP+8]	
004014A4	. E8 34CE0500	CALL XoftSpy_.0045E2D0	
004014A9	. 6A 00	PUSH 0	
004014AB	✓EB 10	JMP SHORT XoftSpy_.004014BD	
004014AD	> 68 94514800	PUSH XoftSpy_.00485194	ASCII "This XoftSpy license has not been registered"
004014B2	. 8D4C24 08	LEA ECX,DWORD PTR SS:[ESP+8]	
004014B6	. E8 22CE0500	CALL XoftSpy_.0045E2D0	
004014BB	. 6A 01	PUSH 1	
004014BD	> 8D8E_EC020000	LEA ECX,DWORD PTR DS:[ESI+2EC]	

AL 값에 따라 보여지는 문자열이 다른걸 알 수 있다. 이전 함수에 의해 AL값이 결정되는것 같아 해당 루틴의 시작 부분에 bp를 걸고 분석해본다.

프로그램 실행하고, **About** 메뉴를 누르자마자 bp가 걸렸다. 왜냐하면 **About**부분을 누르면 등록 여부 메시지가 보이기 때문이다.

004013DD	. 90	NOP	
004013DE	. 90	NOP	
004013DF	. 90	NOP	
004013E0	. 6A FF	PUSH -1	
004013E2	. 68 10774600	PUSH XoftSpy_.00467710	SE handler installation
004013E7	. 64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
004013ED	. 50	PUSH EAX	
004013EE	. 64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
004013F5	. 83EC 28	SUB ESP,28	
004013F8	. 56	PUSH ESI	
004013F9	. 8BF1	MOV ESI,ECX	
004013FB	. 8D4424 1C	LEA EAX,DWORD PTR SS:[ESP+1C]	
004013FF	. 8B4E 1C	MOV ECX,DWORD PTR DS:[ESI+1C]	
00401402	. 50	PUSH EAX	
00401403	. 51	PUSH ECX	
00401404	. FF15 7C054700	CALL DWORD PTR DS:[<&USER32.GetClientRe	pRect hwnd GetClientRect

아까 패치했던 등록관련 루틴에서 등록에 성공해도 실패 문자열을 반환하는 이유는 해당 루틴에서 등록여부를 한번 더 체크하기 때문이다. 즉, 해당 루틴에서 패치해야지 완전한 등록이 된다.

0043500B	. C64424 34 01	MOV BYTE PTR SS:[ESP+34],1	
0043500E	. E8 BBC2FFFF	CALL XoftSpy_.004313A0	
004350E5	. 8B5424 14	MOV EDX,DWORD PTR SS:[ESP+14]	
004350E9	. 83C4 08	ADD ESP,8	
004350EC	. 8B42 F8	MOV EAX,DWORD PTR DS:[EDX-8]	
004350EF	. 85C0	TEST EAX,EAX	
004350F1	. 0F85 B5000000	JNZ XoftSpy_.004351AC	
004350F7	. 8B4424 08	MOV EAX,DWORD PTR SS:[ESP+8]	
004350FB	. 8B48 F8	MOV ECX,DWORD PTR DS:[EAX-8]	
004350FE	. 85C9	TEST ECX,ECX	
00435100	. 0F85 A6000000	JNZ XoftSpy_.004351AC	
00435106	. 6A 00	PUSH 0	
00435108	. 68 389F4800	PUSH XoftSpy_.00489F38	ASCII "User"
0043510D	. 8D4C24 20	LEA ECX,DWORD PTR SS:[ESP+20]	
00435111	. 68 04544800	PUSH XoftSpy_.00485404	ASCII "XoftS"
00435116	. 51	PUSH ECX	
00435117	. E8 84BEFFFF	CALL XoftSpy_.00430FA0	

아까 bp를 걸었던 함수 부분
내부로 들어와 분석을 진행한다.

분석하다보면 첫번째로 분기문이
보이는데, 조건이 충족되어
분기된다.

00435244	. 8B5C24 34	MOV BYTE PTR SS:[ESP+34],BL	
00435248	. E8 63000000	CALL XoftSpy_.004352B0	
0043524D	. 84C0	TEST AL,AL	
0043524F	. 75 04	JNZ SHORT XoftSpy_.00435255	
00435251	. 32DB	XOR BL,BL	
00435253	. EB 02	JMP SHORT XoftSpy_.00435257	
00435255	. B3 01	MOV BL,1	
00435257	. 8D4C24 10	LEA ECX,DWORD PTR SS:[ESP+10]	
0043525B	. C64424 2C 04	MOV BYTE PTR SS:[ESP+2C],4	
00435260	. E8 3B8F0200	CALL XoftSpy_.0045E1A0	
00435265	. 8D4C24 14	LEA ECX,DWORD PTR SS:[ESP+14]	
00435269	. C64424 2C 01	MOV BYTE PTR SS:[ESP+2C],1	
0043526E	. E8 2D8F0200	CALL XoftSpy_.0045E1A0	
00435273	. 8D4C24 08	LEA ECX,DWORD PTR SS:[ESP+8]	
00435277	. C64424 2C 00	MOV BYTE PTR SS:[ESP+2C],0	
0043527C	. E8 1F8F0200	CALL XoftSpy_.0045E1A0	
00435281	. 8D4C24 0C	LEA ECX,DWORD PTR SS:[ESP+C]	
00435285	. C74424 2C FFFF	MOV DWORD PTR SS:[ESP+2C],-1	
0043528D	. E8 0E8F0200	CALL XoftSpy_.0045E1A0	
00435292	. 8B4C24 24	MOV ECX,DWORD PTR SS:[ESP+24]	
00435296	. 8AC3	MOV AL,BL	
00435298	. 5E	POP ESI	
00435299	. 64:890D 000000	MOV DWORD PTR FS:[0],ECX	
004352A0	. 5B	POP EBX	
004352A1	. 83C4 28	ADD ESP,28	
004352A4	. C3	RETN	
004352A5	. 90	NOP	

계속 분석하다보면 AL 부분을
결정짓는 코드를 볼 수 있다.
BL에 1을 저장하고 이후에 AL,
BL을 통해 AL에 최종적으로 1이
들어감 등록 성공 메시지를 볼 수
있다.

00435244	. 885C24 34	MOV BYTE PTR SS:[ESP+34],BL
00435248	. E8 63000000	CALL XoftSpy_.004352B0
0043524D	. 84C0	TEST AL,AL
0043524F	. 75 04	JNZ SHORT XoftSpy_.00435255
00435251	. 32DB	XOR BL,BL
00435253	. EB 02	JMP SHORT XoftSpy_.00435257
00435255	. B3 01	MOV BL,1
00435257	. 8D4C24 10	LEA ECX,DWORD PTR SS:[ESP+10]
0043525B	. C64424 2C 04	MOV BYTE PTR SS:[ESP+2C],4
00435260	. E8 3B8F0200	CALL XoftSpy_.0045E1A0
00435265	. 8D4C24 14	LEA ECX,DWORD PTR SS:[ESP+14]
00435269	. C64424 2C 01	MOV BYTE PTR SS:[ESP+2C],1
0043526E	. E8 2D8F0200	CALL XoftSpy_.0045E1A0
00435273	. 8D4C24 08	LEA ECX,DWORD PTR SS:[ESP+8]
00435277	. C64424 2C 00	MOV BYTE PTR SS:[ESP+2C],0
0043527C	. E8 1F8F0200	CALL XoftSpy_.0045E1A0
00435281	. 8D4C24 0C	LEA ECX,DWORD PTR SS:[ESP+C]
00435285	. C74424 2C FFFF	MOV DWORD PTR SS:[ESP+2C],-1
0043528D	. E8 0E8F0200	CALL XoftSpy_.0045E1A0
00435292	. 8B4C24 24	MOV ECX,DWORD PTR SS:[ESP+24]
00435296	. 8AC3	MOV AL,BL
00435298	. 5E	POP ESI
00435299	. 64:890D 000000	MOV DWORD PTR FS:[0],ECX
004352A0	. 5B	POP EBX
004352A1	. 83C4 28	ADD ESP,28
004352A4	. C3	RETN
004352A5	. 90	NOP

하지만 435255부분으로 분기해야 BL에 1이 들어가는데, 조건이 충족되지 않아 BL에 0이 들어가게 된다.

그대로 진행하면 435257로 분기하게 되고, MOV AL, BL 명령을 통해 AL 결국 0이 들어오게 된다. 그래서 이부분을 MOV AL, 1로 패치해주면 될것같다.

00401484	. E8 17F02000	CALL XoftSpy_.00431340	
00401489	. 8B8E C0000000	MOV ECX,DWORD PTR DS:[ESI+C0]	
0040148F	. 83C4 08	ADD ESP,8	
00401493	. E8 093C0300	CALL XoftSpy_.004350A0	
00401497	. 84C0	TEST AL,AL	
00401499	. 74 12	JE SHORT XoftSpy_.004014AD	
0040149B	. 68 C4514800	PUSH XoftSpy_.004851C4	ASCII "This license of XoftSpy has been registered"
004014A0	. 8D4C24 08	LEA ECX,DWORD PTR SS:[ESP+8]	
004014A4	. E8 34CE0500	CALL XoftSpy_.0045E2D0	
004014A9	. 60 00	PUSH 0	
004014AB	. EB 10	JMP SHORT XoftSpy_.004014BD	

다시 코드로 돌아오면 AL이 1이기 때문에 JE또한 분기되지 않고, 결국 성공 메시지를 반환하게 된다.

This license of XoftSpy has been registered

<http://www.paretologic.com/xoftspy>

<http://www.paretologic.com/support>

Copyright ?2004 ParetoLogic Inc.

All rights reserved.

[Terms of use](#)