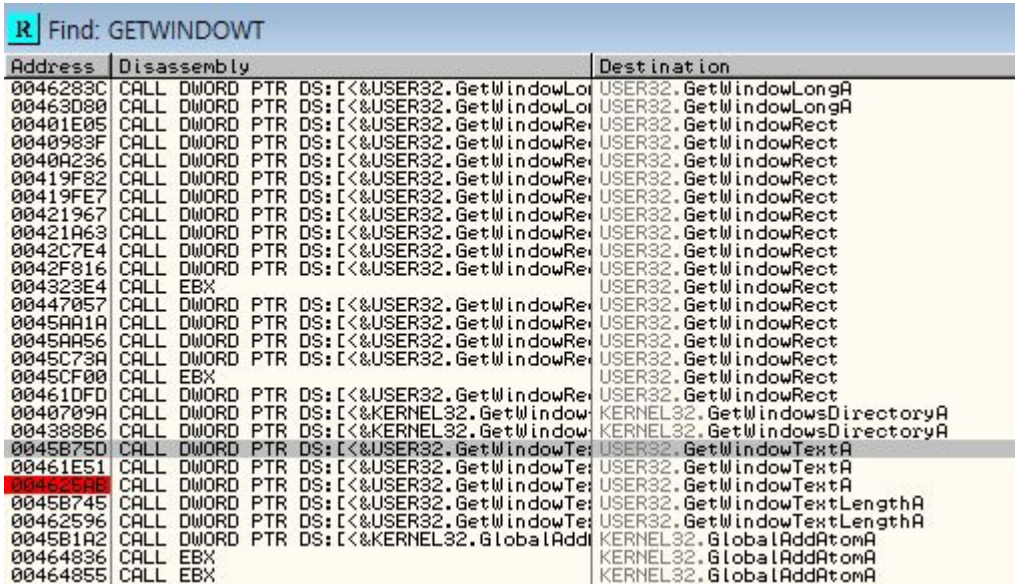
xoftspy라는 프로그램은 윈도우용 스파이웨어 방지 프로그램으로 USB 드라이브에서 실행되는 일종의 백신 프로그램이다.

이번 프로그램도 등록 우회 하는게 목적이다.

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



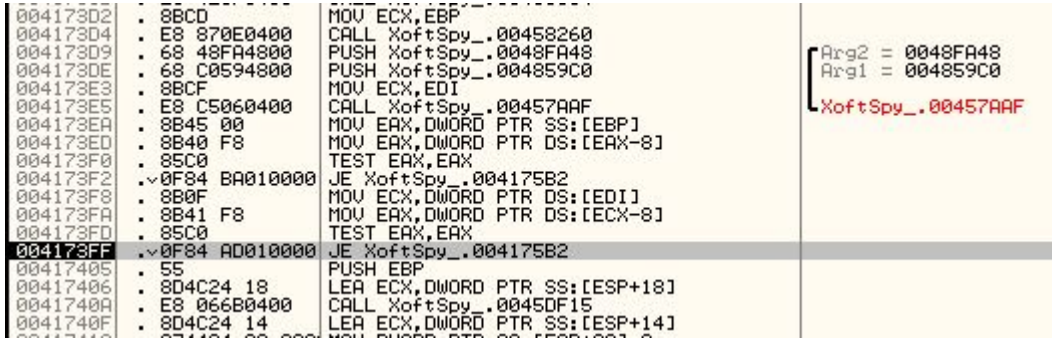
이번 리버싱의 목적은 all intermodules calls를 이용해 전처럼 문자열을 검색해 오류를 패치하는 것이 아닌 문자열이 없을 시 해당 프로그램에서 호출하는 API를 이용해 패치하는것이 목적이다.

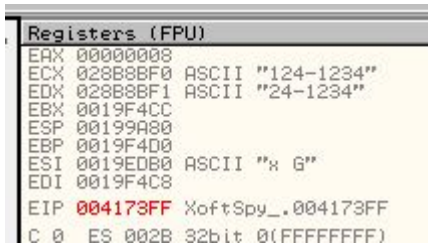
ollydbg all intermodules calls에서 GetWindowTextA를 검색해 해당 함수를 호출하는 모든 부분에 bp를 건다.

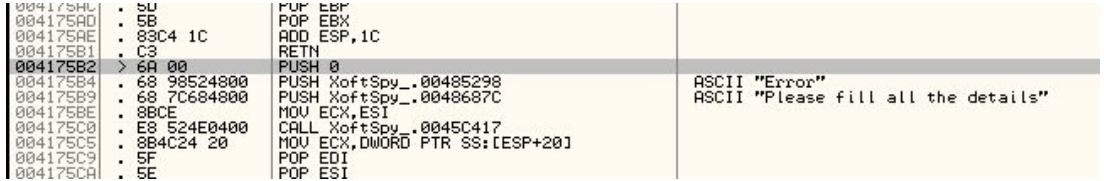
코드를 입력하고 OK 버튼을 누르면 해당 영역에서 멈추게 된다.

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

등록 코드 앞에 뒤에 그리고 이름까지 총 세번 반복하면서 읽어온다.



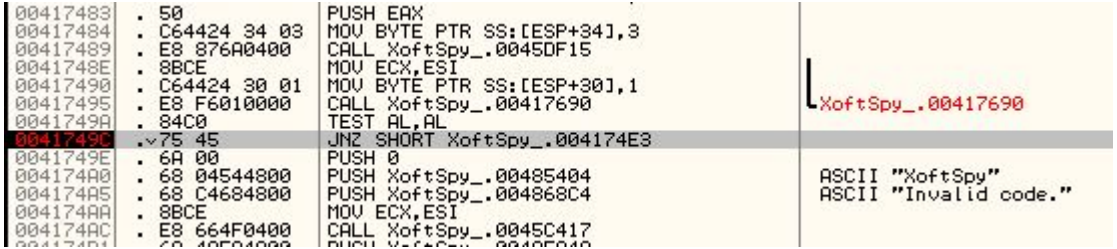


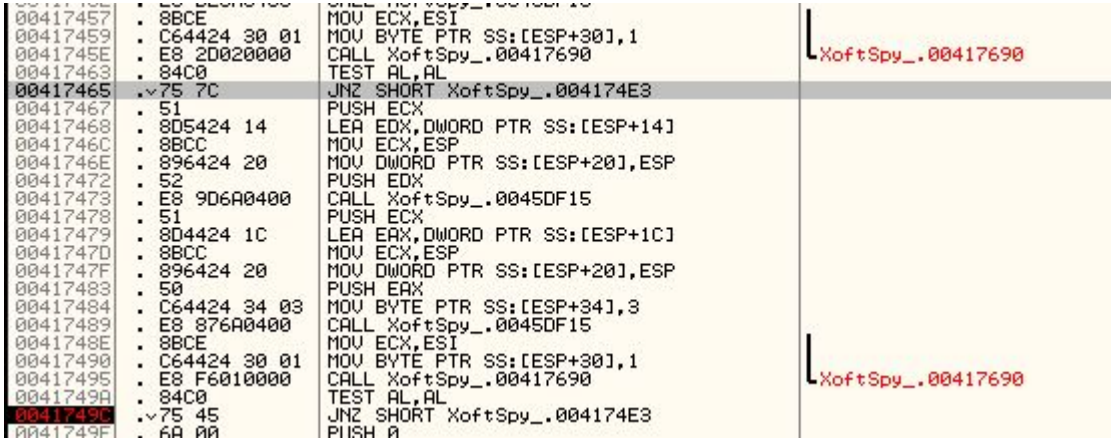


분석하다 보면 해당 분기문을
만나게 된다.

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.





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

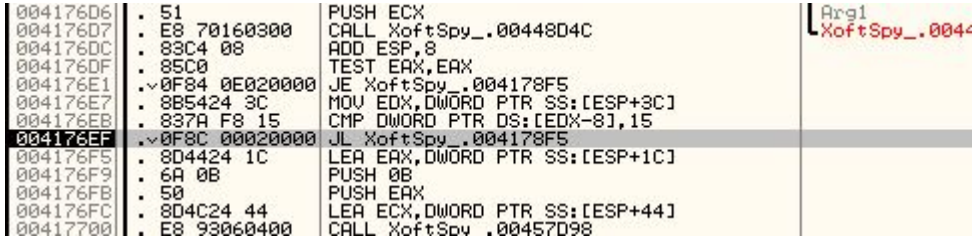
분기문 이전에 똑같은 함수를 두 번 호출해 AL값을 결정짓는걸 알 수 있다. 첫번째 함수에 bp를 걸고 분석해봐야 할 것 같다.

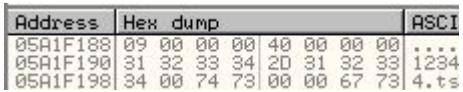
```
00417903  . 8D4C24 3C      LEA ECX,DWORD PTR SS:[ESP+3C]
00417907  . C74424 30 FFFF  MOV DWORD PTR SS:[ESP+30],-1
0041790F  . E8 8C680400     CALL XoftSpy_.0045E1A0
00417914  . 8B4C24 28       MOV ECX,DWORD PTR SS:[ESP+28]
00417918  . 5F              POP EDI
00417919  . 5E              POP ESI
0041791A  . 32C0            XOR AL,AL
0041791C  . 5B              POP EBX
0041791D  . 64:890D 00000(  MOV DWORD PTR FS:[0],ECX
00417924  . 83C4 28         ADD ESP,28
00417927  L. C2 0800        RETN 8
0041792A    90              NOP
```

```
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 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]
```

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

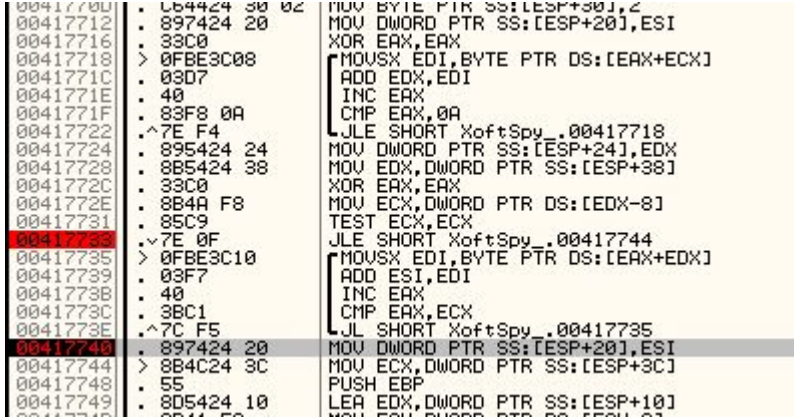
그래서 이전 명령어를 살펴봤는데, 4178AC 부분 전에 보면 AL값을 설정하는 부분이 나온다. 이 부분에서 AL을 1로 설정하므로 해당 부분이 호출되게 해야된다. 어떤 부분에서 해당 부분을 호출하는지 살펴본다.





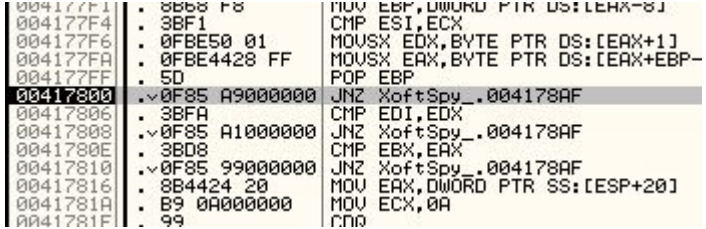
프로그램을 살펴보니 어떤 부분도 AL, 1부분으로 분기하지 않았다. 그래서 살펴보니 해당 부분에서 AL, 0으로 점프를 시도하는걸 볼 수 있다. 분기하는 이유을 분석해보니 CMP PTR DS:[EDX-8], 15부분에서 비교하고 이 값보다 작을 경우 분기를 시도한다.

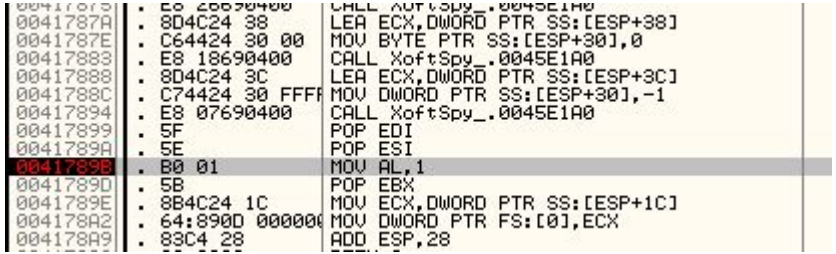
EDX-8부분을 살펴보면 9라는 값이 들어가 있는데, 이 값은 내가 입력한 코드의 길이를 나타내는것을 알 수 있다. 즉, 코드 길이는 15이상이어야 한다.



분기하는 부분의 S플래그를 조작해 분기하지 않도록 한다.

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



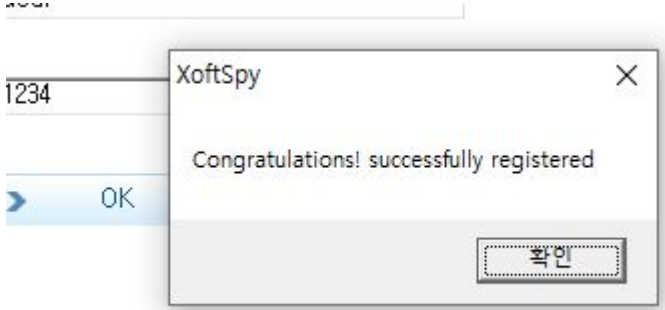


또 내려가다보면 세개의 분기문이 나오고 세개 모두 다 같은 곳으로 분기한다.

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

플래그를 변경해가며 진행하다보면, AL, 1부분에 도달하게 된다.

```
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  .v75 7C           JNZ SHORT XoftSpy_.004174E3
00417467  . 51              PUSH ECX
00417468  . 8D5424 14       LEA EDX,DWORD PTR SS:[ESP+14]
0041746C  . 8BCC            MOV ECX,ESP
```

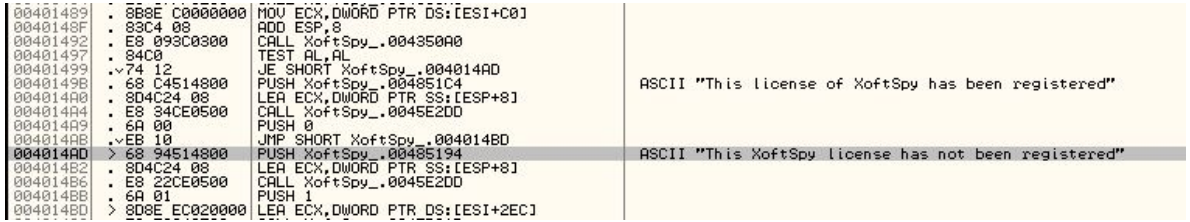


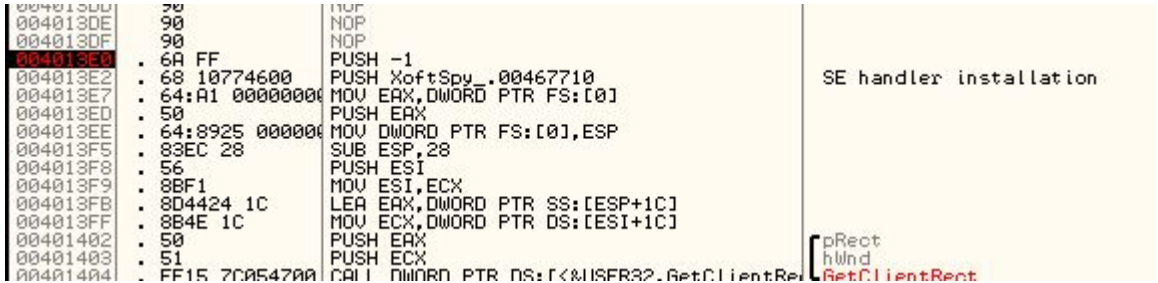
AL 부분을 조작했으니 아까 분기 못했던 JNZ 4174E3도 분기가 된다.

이렇게 성공 메시지를 볼 수 있다.

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

이번에는 API가 아닌 문자열로 검색한다.



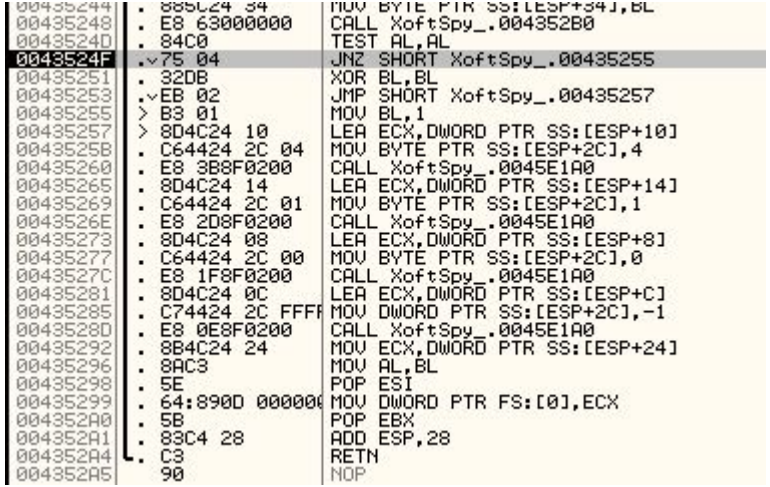


AL 값에 따라 보여지는 문자열이 다른걸 알 수 있다. 이전 함수에 의해 AL값이 결정되는것 같아 해당 루틴의 시작 부분에 bp를 걸고 분석해본다.

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

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

```
004350DB  . C64424 34 01  MOV BYTE PTR SS:[ESP+34],1
004350E0  . 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  .v0F85 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  .v0F85 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 "Xoft
00435116  . 51            PUSH ECX
00435117  . E8 84BEFFFF   CALL XoftSpy_.00430FA0
```



```
00435244  . 885C24 34     MOV BYTE PTR SS:[ESP+34],BL
00435248  . E8 63000000   CALL XoftSpy_.004352B0
0043524D  . 84C0          TEST AL,AL
0043524F  .v75 04         JNZ SHORT XoftSpy_.00435255
00435251  . 32DB          XOR BL,BL
00435253  .vEB 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  L. C3           RETN
004352A5    90            NOP
```

아까 bp를 걸었던 함수 부분 내부로 들어와 분석을 진행한다.

분석하다보면 첫번째로 분기문이 보이는데, 조건이 충족되어 분기된다.

계속 분석하다보면 AL 부분을 결정짓는 코드를 볼 수 있다. BL에 1을 저장하고 이후에 AL, BL을 통해 AL에 최종적으로 1이 들어가 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 .v75 04          JNZ SHORT XoftSpy_.00435255
00435251 . 32DB           XOR BL,BL
00435253 .vEB 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 L. C3            RETN
004352A5   90             NOP
```

```
00401484 . E8 17FF0200    CALL XoftSpy_.004313A0
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 .v74 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_.0045E2DD
004014A9 . 6A 00          PUSH 0
004014AB .vEB 10          JMP SHORT XoftSpy_.004014BD
```

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.

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