악성코드 분석 보고서

(sand-reversingwithlena-tutorials)

2025.07.14

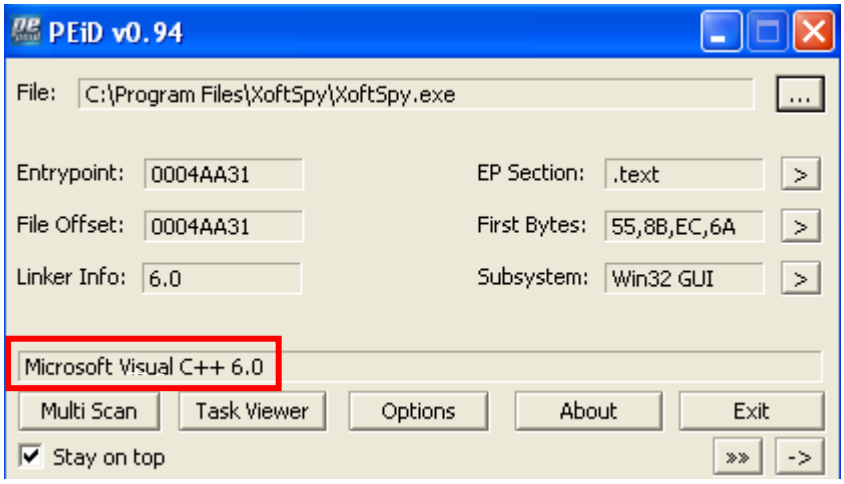
13

1. 문제

레지스터 키가 아직 등록되지 않았다는 내용

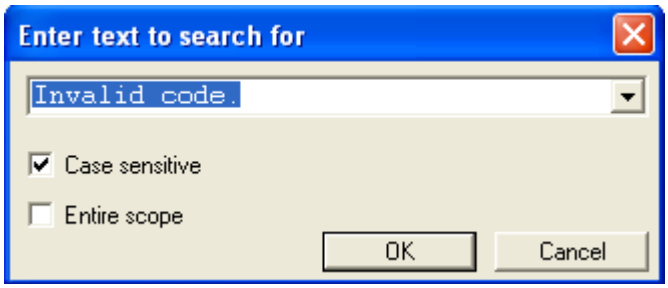
키 등록해야함

2-1. 해결 방법

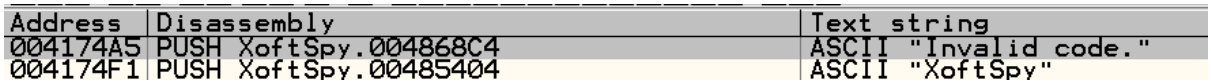


PEID로 확인 시 C++로 만든 파일이라는 것을 알 수 있다.
키가 틀린 경우 'Invalid code.'라고 뜨는데

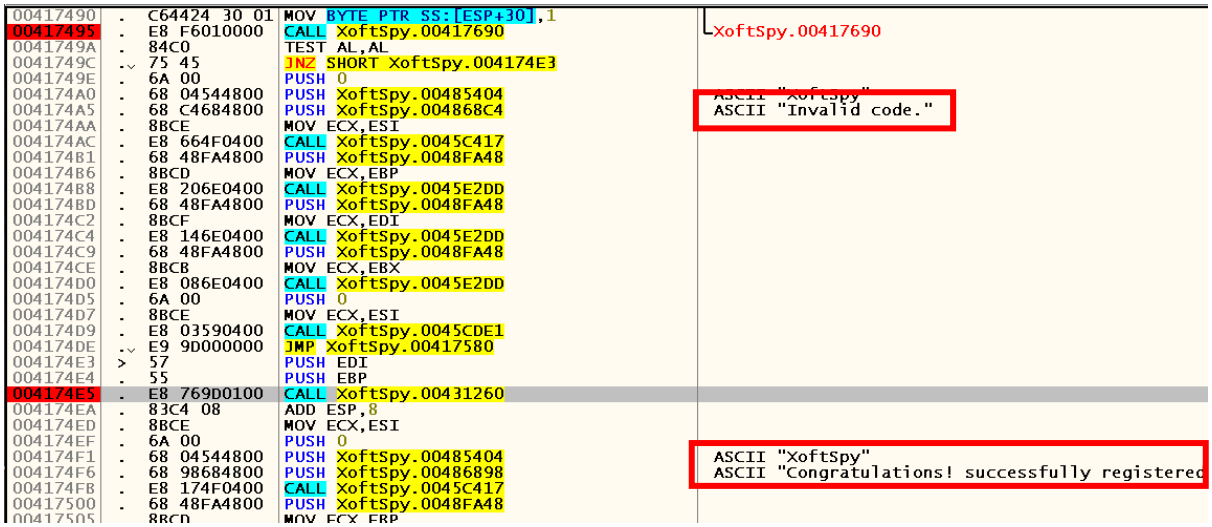
우선 키가 등록 성공할 경우 'This XoftSpy license has not been registered' 이게 바뀔 수도 있으니까 키부터 찾아준다.



Search for > All referenced text strings 눌러서 Invalid code. 해당 구문을 찾아준다.

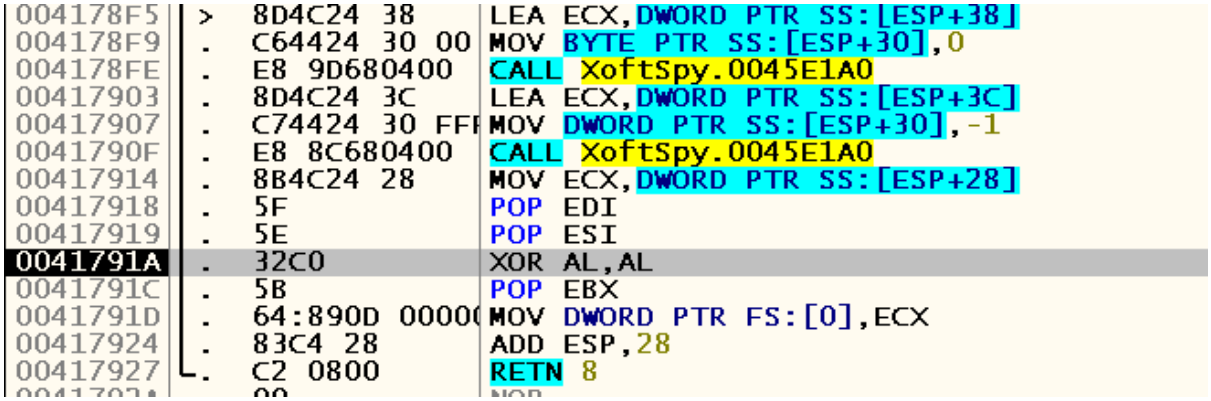


나오는 걸 볼 수 있고 들어가서 주변 분기문을 찾아주고 브레이크를 걸어준다.



분기문을 찾고 키 등록 성공했을 때 구문도 찾아서 브레이크 걸어준다.

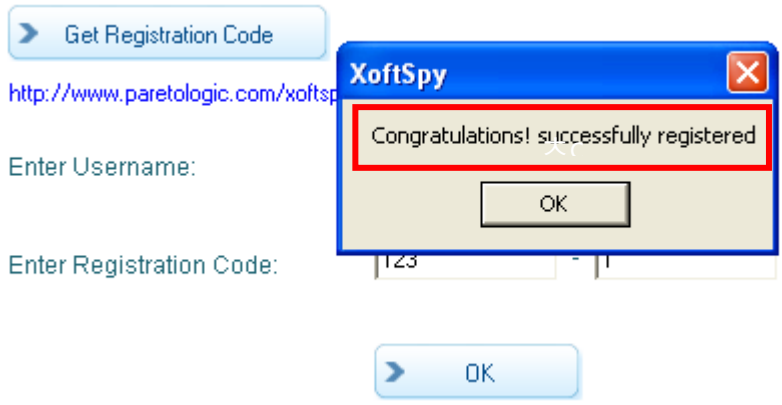
실행을 하고 임시로 키 입력하면 0x00417495에서 멈추고 0x0041749C에서 점프를 하게 만들어 줘야하니까 AL 값이 0이 아닌 값이 나와야한다.



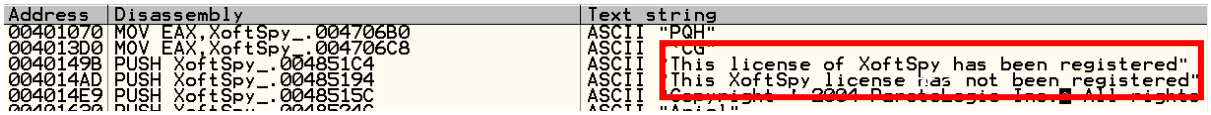
0x0041749 들어가서 보면 XOR AL, AL에서 0으로 만드는 것을 볼 수 있다.

이 부분을 MOV AL, 1로 바꿔주고 저장해준다.

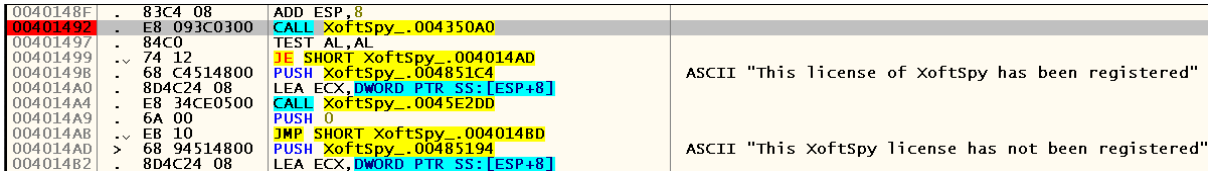
바꾼 파일에 들어가서 실행해본다.



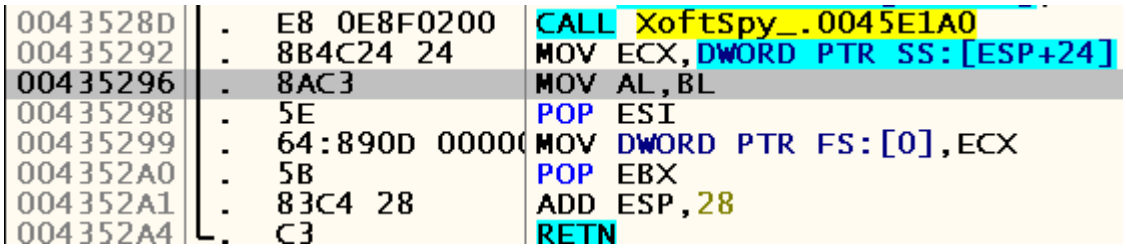
등록에 성공했다는 창이 발생했지만 다시 보면 아직 등록하지 않았다는 구문은 그대로 있다. 그럼 이 구문을 OllyDbg에서 찾아준다.



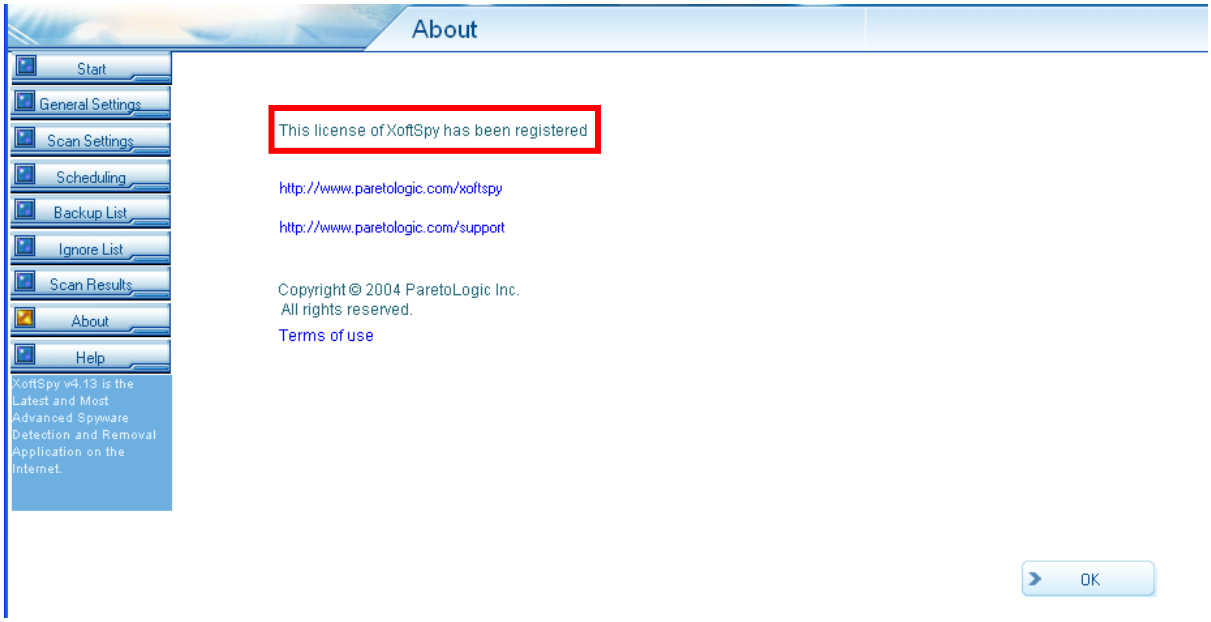
이렇게 등록에 성공했다는 문자열과 성공하지 못했다는 문자열이 같이 있는 걸 볼 수 있다.



들어가서 분기문을 찾아서 살펴보면 0x00401492에서 AL값을 리턴하는데 AL값이 0만 아니면 된다. 그래서 JE를 NOP으로 바꿔주거나 AL = 1로 바꿔주면 되는데 후자로 수정할 해볼 것이다.



0x00401492로 들어가서 보면 AL이 BL의 값을 받아오는 것을 볼 수 있다. 이거를 MOV AL, 1로 바꿔서 저장해준다.



실행해보면 등록했다는 구문으로 바뀌고 레지스터 키 등록하는 버튼이 없어진 걸 볼 수 있다.

문제 해결!

-> 하지만 lena의 영상을 보고 풀 경우 다른 방식으로 풀 수 있다.

2-2. Lena 방식

Lena는 window API를 이용해서 풀었다.

-> 이 방식으로 풀어야 좀 더 API에 대해 많이 알 수 있다고 생각함.

- Lena가 설명한 프로그램과 커널의 상호작용 API

1. DialogBoxes -DialogBoxParamA -GetDlgItem -GetDlgItemInt -GetDlgItemTextA -GetDlgItemTextA -GetWindowTextA -GetWindowWord	**2. MessageBoxes** -MessageBeep -MessageBoxA -MessageBoxExA -SendMessageA -SendDlgItemMessageA	**3. Registry Access** -RegCreateKeyA -RegDeleteKeyA -RegQueryValueA -RegQueryValueExA -RegCloseKeyA -RegOpenKeyA
4. Read/Write File -ReadFile -WriteFile -CreateFileA	**5. Reading data from (*.ini)** -GetPrivateProfileStringA -GetPrivateProfileIntA -WritePrivateProfileStringA	**6. Reading data (other)** -LoadStringA -lstrcmpA -MultiByteToWideChar -WideCharToMultiByte -wsprintfA
7. Time and Date -GetFileTime -GetLocalTime -GetSystemTime -GetSystemTimeAsFileTime -SetTimer -SystemTimeToFileTime	**8. window** -CreateWindowExA -ShowWindow -UpdateWindow	**9. MessageBoxText** -SendDlgItemMessageA -SendMessageA -SetDlgItemTextA -SetWindowTextA

여기서 Lena는 등록 방식을 분석할 때, 다음 API 호출들 중 하나(또는 전부)에 브레이크 포인트를 건다. ('사용자 입력, 설정 파일/레지스트리 접근, 문자열 비교'에 관여하는 주요 API)

- GetDlgItemTextA : 텍스트 박스에서 사용자 입력 읽어옴

- GetWindowTextA : 윈도우에서 텍스트 읽어옴

- lstrcmpA : 문자열 비교 (예: 입력한 시리얼과 내부 값 비교)

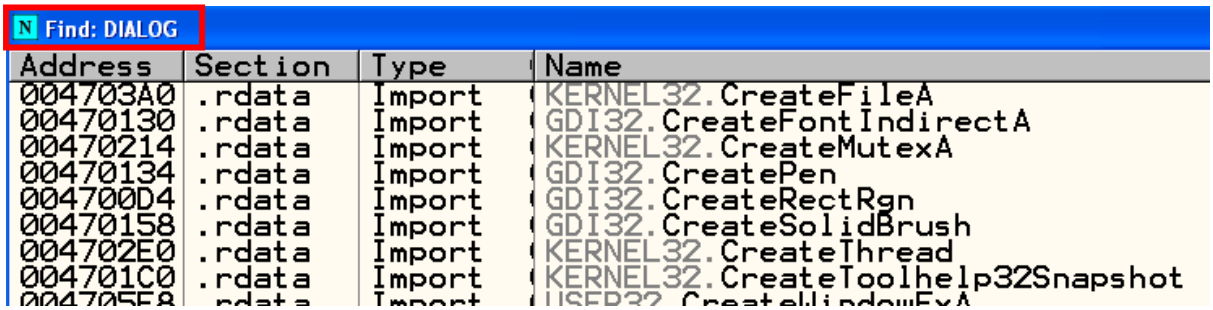
- GetPrivateProfileStringA : INI 파일에서 문자열 읽기

- GetPrivateProfileIntA : INI 파일에서 정수 읽기

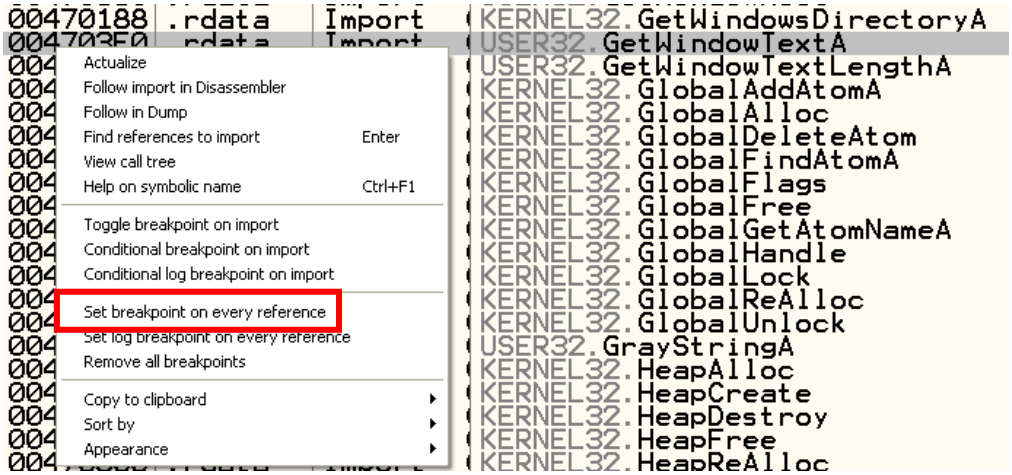
- RegQueryValueExA : 레지스트리 값 읽기

- WritePrivateProfileStringA : INI 파일에 문자열 쓰기

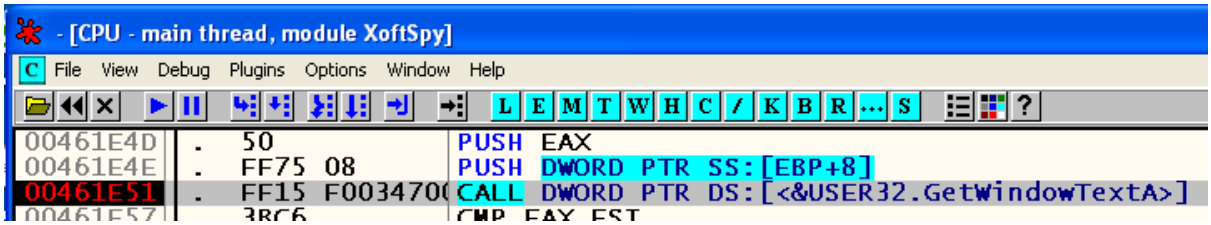
- WritePrivateProfileIntA : INI 파일에 정수 쓰기



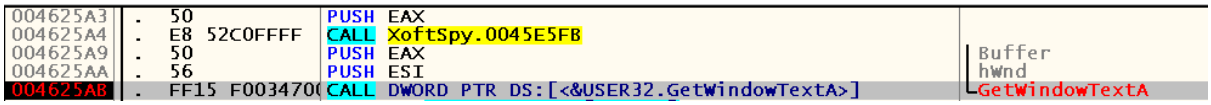
OllyDbg에서 Windows API 목록을 보면 Dialog 기반은 안사용하는 것을 볼 수 있다.



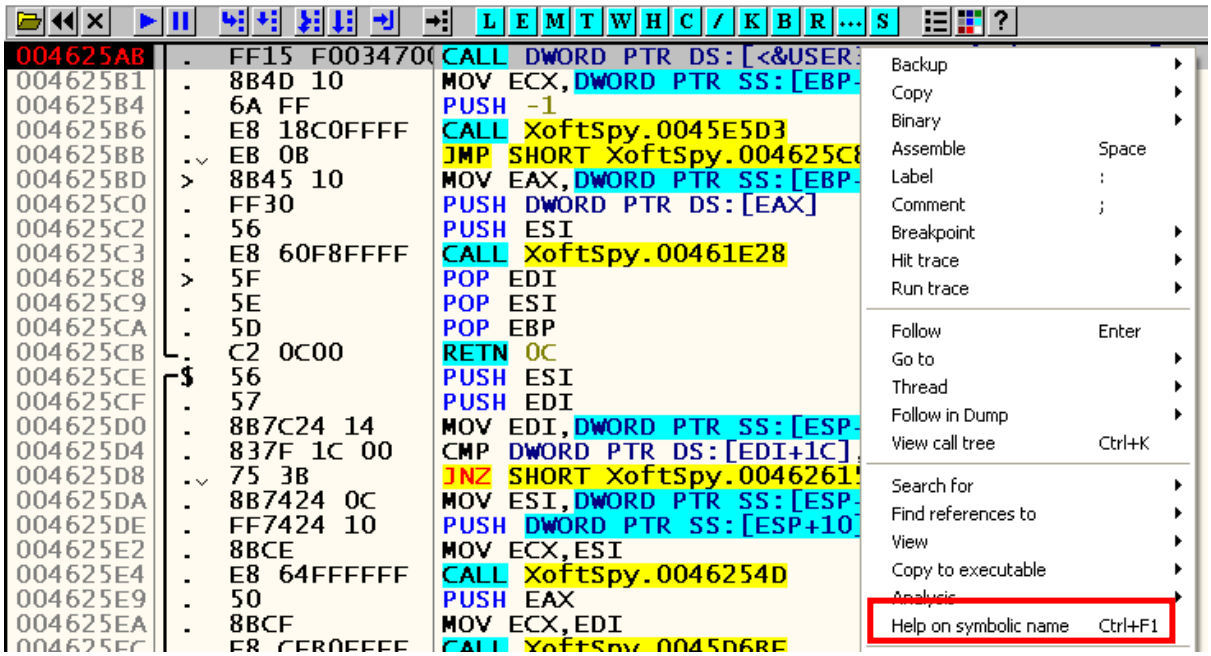
그럼 Window기반을 사용하니까 GetWindowTextA를 찾아서 이 모듈을 사용하는 곳에 다 브레이크를 걸어준다.



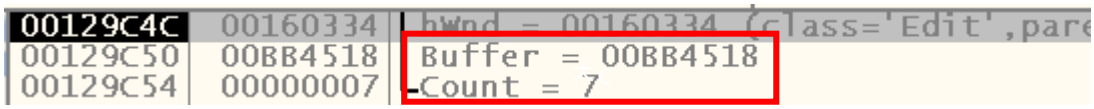
'F9'를 눌러 실행해주면 0x00461E51에서 멈추는데 여기는 실행도 하기 전에 브레이크 걸린 곳이라서 브레이크를 풀어주고 다시 실행한다.



키를 입력하고 실행시키면 해당 구문에 멈춘다.



이 API가 하는 일을 win32.hlp 문서를 참고해서 확인할거다. 왜냐하면 등록과정에 관련되어 있을거라고 추측하기 때문이다.



읽고 스택을 보면 버퍼랑 카운트가 보이는데 입력한 글자에서 7글자만 가져온다는 뜻이다.

0x004625AB가 총 3번 반복되는데 두 번째 비밀번호 > 첫 번째 비밀번호 > 이름 순으로 체크한다.

```
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   .v  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   .v  0F84 AD010000  JE XoftSpy.004175B2
```

여기서 JE분기문이 두 번이나 반복되는데 첫 번째 EAX는 이름의 개수를 세고 없으면 0x004175B2로 가서 칸을 채워달라는 창이 발생하는 구간이고

두 번째 EAX는 전체 비밀번호의 개수를 세는 구간이고 똑같이 없으면 0x004175B2로 가서 칸을 채워달라는 창이 발생하는 구간으로 넘어간다.

```
00417452   .   E8 BE6A0400   CALL XoftSpy.0045DF15
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   .v  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   .v  75 45         JNZ SHORT XoftSpy.004174E3
```

이 부분이 2번 반복되는데 이름과 비밀번호가 맞는지 비교하는 부분으로 생각된다.

둘 다 AL = 1이 나와야 JNZ 분기문에서 0x004174E3으로 가는데 이 부분이 값이 맞다고 나오는 부분이다. 근데 진행해보면 AL = 0이 나오는데 이 부분을 바꾸기 위해서는 0x00417690함수를 들어가서 살펴봐야한다.

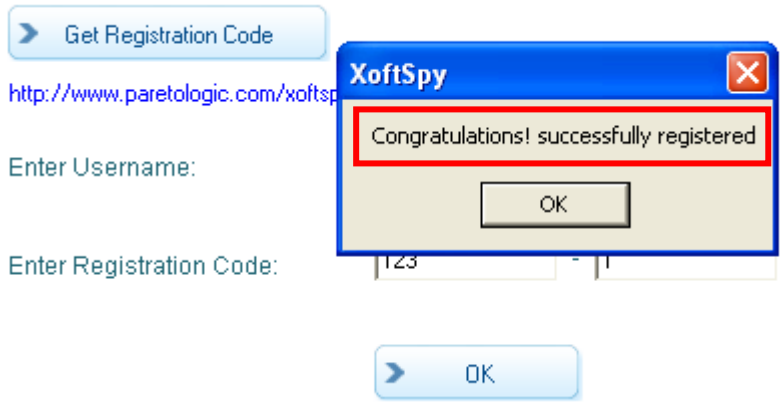
```
00417894   .   E8 07690400   CALL XoftSpy.0045E1A0
00417899   .   5F            POP EDI
0041789A   .   5E            POP ESI
0041789B   .   B0 01         MOV AL,1
0041789D   .   5B            POP EBX
0041789E   .   8B4C24 1C     MOV ECX,DWORD PTR SS:[ESP+1C]
004178A2   .   64:890D 00000 MOV DWORD PTR FS:[0],ECX
004178A9   .   83C4 28       ADD ESP,28
004178AC   .   C2 0800       RETN 8
```

```
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 0000( MOV DWORD PTR FS:[0],ECX
00417924   .  83C4 28       ADD ESP,28
00417927   L. C2 0800       RETN 8
```

return값을 찾다보면 총 2개가 나오는데 첫 번째 return값을 보면 MOV AL, 1이 있는데 AL = 1로 바꿔주는 걸로 보면 입력한 값이 성공했을 때 저기로 가는 것 같고

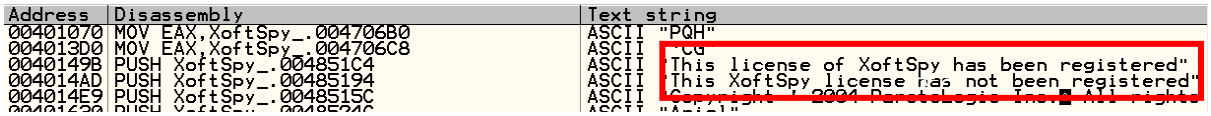
두 번째 return값을 보면 XOR AL, AL이 있는데 AL = 0으로 바꿔주는 걸로 보면 실패할 때 저기로 가는 것 같다.

그럼 실패해도 AL = 1이 나오게 해야하니까 0x0041791A의 XOR AL, AL을 MOV AL, 1로 바꿔주고 저장한다.

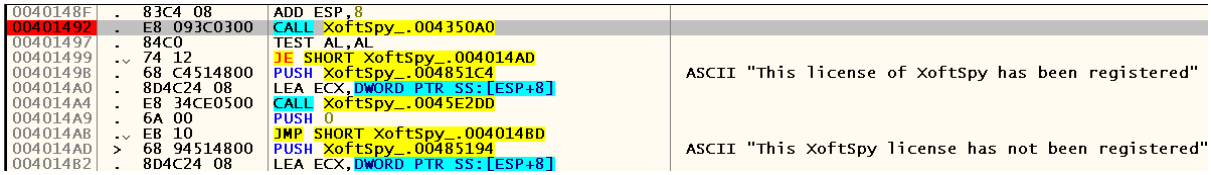


값을 입력해보면 무조건 성공했다는 창이 나올 것이다.

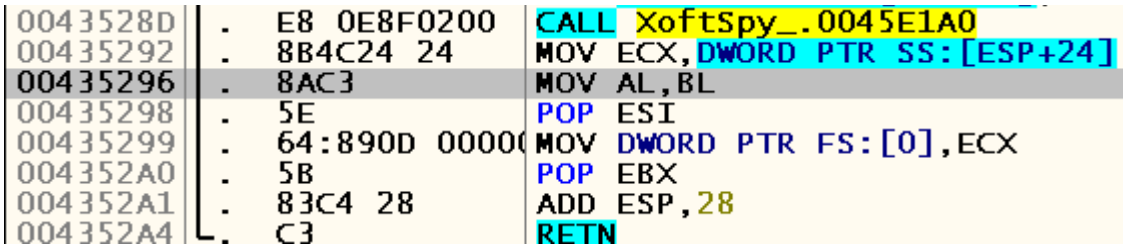
등록에 성공했다는 창이 발생했지만 다시 보면 아직 등록하지 않았다는 구문은 그대로 있다. 그럼 이 구문을 OllyDbg에서 찾아준다.



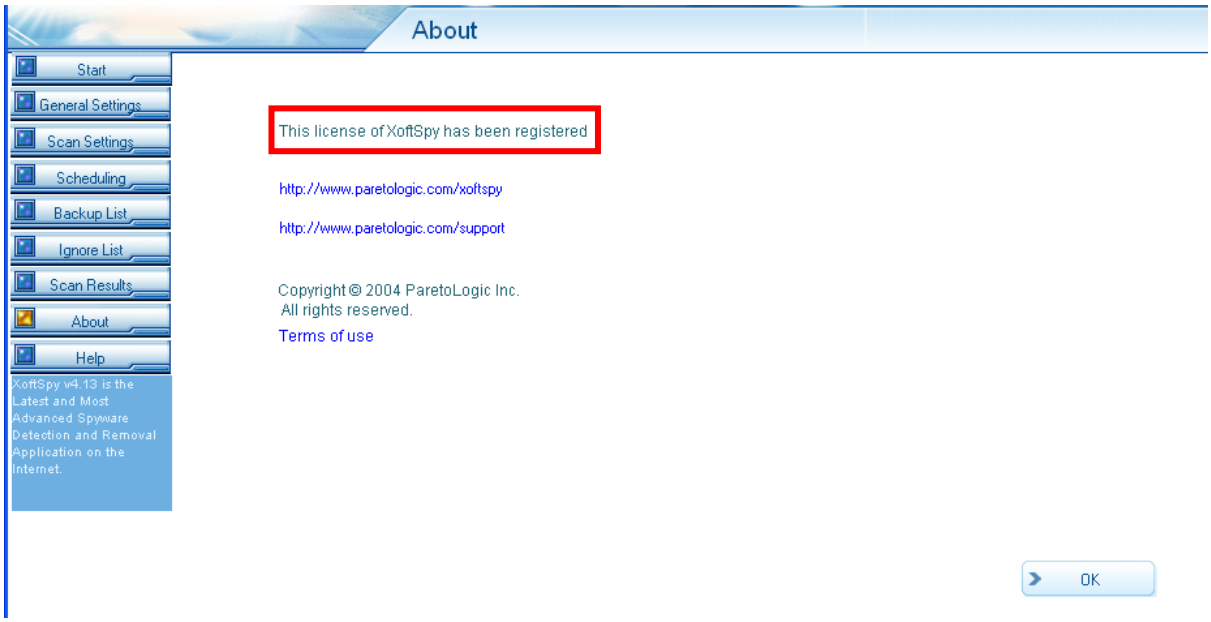
이렇게 등록에 성공했다는 문자열과 성공하지 못했다는 문자열이 같이 있는 걸 볼 수 있다.



들어가서 분기문을 찾아서 살펴보면 0x00401492에서 AL값을 리턴하는데 AL값이 0만 아니면 된다. 그래서 JE를 NOP으로 바꿔주거나 AL = 1로 바꿔주면 되는데 후자로 수정할 해볼 것이다.



0x00401492로 들어가서 보면 AL이 BL의 값을 받아오는 것을 볼 수 있다. 이거를 MOV AL, 1로 바꿔서 저장해준다.



실행해보면 등록했다는 구문으로 바뀌고 레지스터 키 등록하는 버튼이 없어진 걸 볼 수 있다.

전체적인 흐름은 같지만 lena는 초반에 API를 이용해서 푼 걸 볼 수 있다.