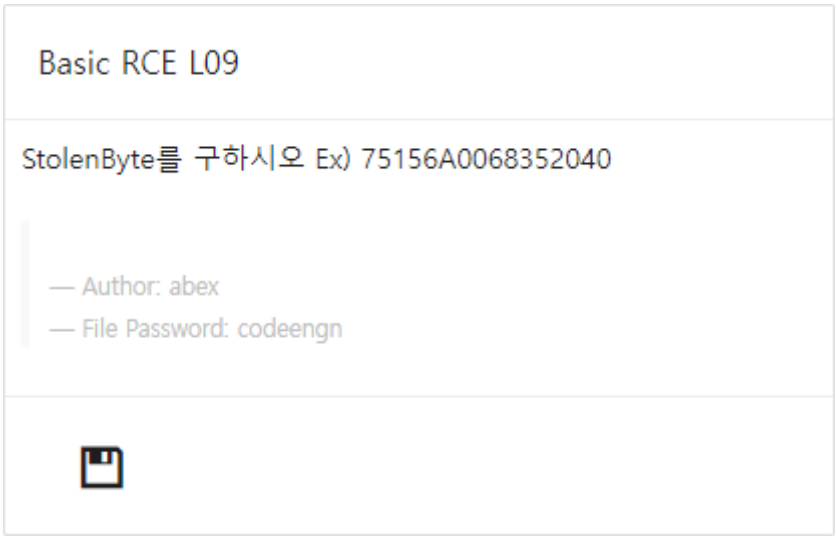
codeengn-basic-L09 풀이

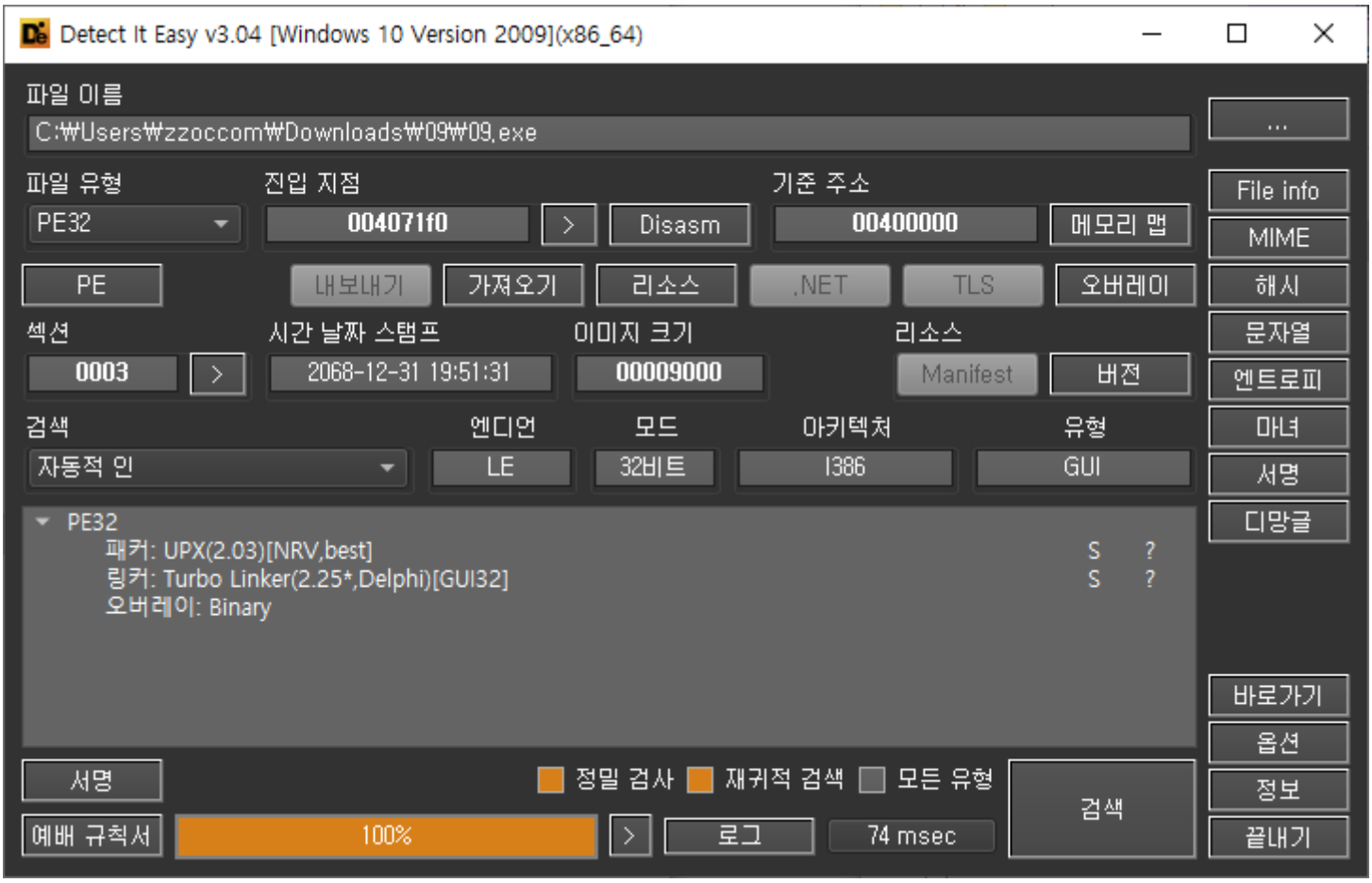
리버싱 문제풀이 / Wonlf / 2022. 3. 27. 15:04



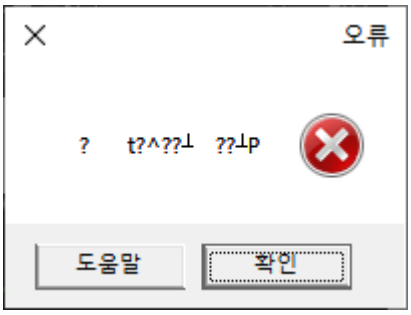
문제는 **StolenByte**를 원하고 있다.

여기서 **StolenByte**란,
직역하면 훔친 바이트란 뜻으로 프로그램의 한 부분의 코드를 훔쳐내어 다른 부분으로 옮겨진 코드를 말한다.
주로 옮겨지는 코드는 엔트리 포인트 위의 몇 개의 옮겨진 코드들이며 OEP주소로 점프하기 전에 위치에서 PUSH 된다. 이러한 StolenByte는 주로 패커가 프로그램을 패킹할 때 볼 수 있는데 이렇게 옮겨진 코드들은 할당된 메모리 공간에서 실행 된다. 이 때문에 패킹된 프로세스가 덤프될 때 **StolenByte**를 복구하지 못하면 프로그램은 정상적으로 작동하지 못하게 된다.
요약 하자면 **StolenByte**란 패커가 위치를 이동시킨 코드로써 보호된 프로그램의 코드의 윗부분이다.
더 간단하게 말하자면 POPAD랑 OEP사이에 있는 연속된 PUSH이다.

그래서 문제를 처음 보게 되면



UPX로 패킹이 되어 있는데 이것을 프로그램을 통해 언패킹 하고 실행을 하게 되면,



이런식으로 파일이 깨져서 실행되게 된다. 위에서 확인 했던 것처럼 패커가 **StolenByte**를 복구하지 못했기 때문에 이렇게 된 것이다. 이것이 원본 코드 중 일부를 별도의 영역에서 실행하게 하여 OEP의 위치를 다른 위치로 가장하는 기법인 **StolenByte**이다. 이제 디버깅을 시작해보겠다.

```
00401000    90          nop                                 EntryPoint
00401001    90          nop
00401002    90          nop
00401003    90          nop
00401004    90          nop
00401005    90          nop
00401006    90          nop
00401007    90          nop
00401008    90          nop
00401009    90          nop
0040100A    90          nop
0040100B    90          nop
0040100C    6A 00       push 0
0040100E    E8 8C000000 call <JMP.&MessageBoxA>
```

언패킹한 파일

이런식으로 언패커를 활용하여 프로그램을 패킹 해제를 하고 디버거로 열어보면 Messagebox가 호출이 되는데 인자값으로 아무것도 들어가지 않으니 상단에서 봤던 오류를 보게 되었다. 다시 패킹된 파일을 열어보겠다.

```
0040736D    61          popad
0040736E    6A 00       push 0
00407370    68 00204000 push 09.402000                      402000:"abex' 3rd crackme"
00407375    68 12204000 push 09.402012                      402012:"Click OK to check for the keyfile."
0040737A    8D4424 80   lea eax,dword ptr ss:[esp-80]
0040737E    6A 00       push 0
00407380    39C4        cmp esp,eax
00407382  ^ 75 FA       jne 09.40737E
00407384    83EC 80     sub esp,FFFFFF80
00407387  ^ E9 809CFFFF jmp 09.40100C
0040738C    0000        add byte ptr ds:[eax],al
0040738E    0000        add byte ptr ds:[eax],al
00407390    0000        add byte ptr ds:[eax],al
```

패킹된 파일

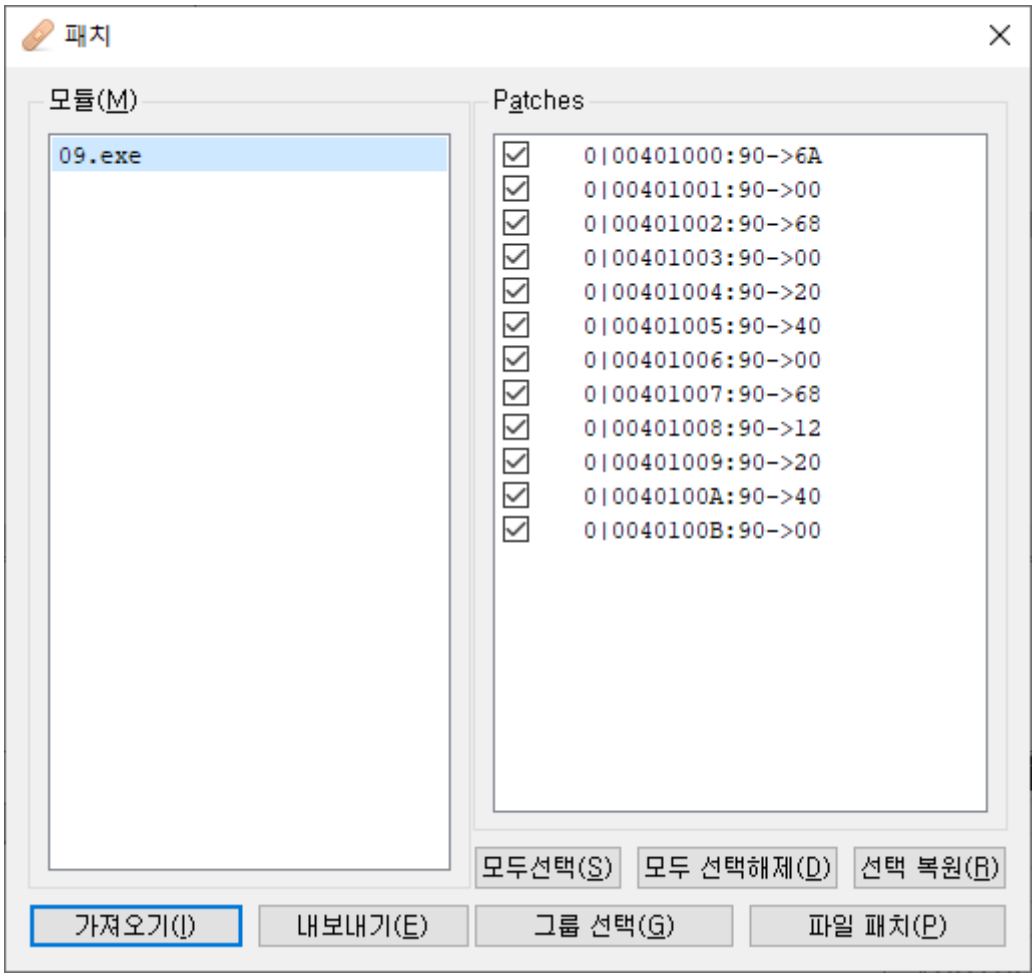
이런식으로 프로그램이 언패킹 작업을 진행하다가 OEP로 가는 구문 위에 push구문 3개가 보일 것이다. 원래 메시지 박스에 출력되어야 하는 문자열들이 OEP로 가기 직전 스택에 들어가는 모습니다. 이런식으로 특정 구문을 다른 메모리 주소에 훔쳐와 저장해놓는 디버깅 방어 기법을 StolenByte라고 한다.
이 push구문 3개를 원래 있던 자리인 언패킹한 파일 EP에 넣어주겠다.

```
00401000    6A 00       push 0                              EntryPoint
00401002    68 00204000 push 09.402000                      402000:"abex' 3rd crackme"
00401007    68 12204000 push 09.402012                      402012:"Click OK to check for the keyfile."
0040100C    6A 00       push 0
0040100E    E8 8C000000 call <JMP.&MessageBoxA>
00401013    6A 00       push 0
00401015    68 80000000 push 80
```

수정한 패킹해제한 파일

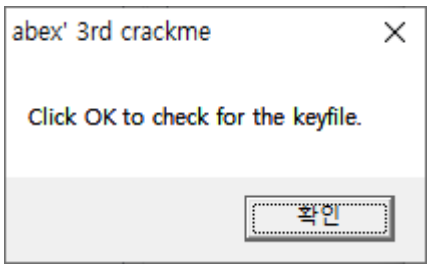
이렇게 NOP로 되어 있던 부분에 Stolenbyte에 있던 구문 3개를 추가시켜주었다.

구문을 추가 했으면 구문을 추가한 버전의 파일을 새로 만들어야한다.



CTRL + P 또는 [파일] - [패치]를 통해 저장 할 수 있다.

패치한 파일을 실행시켜보면,



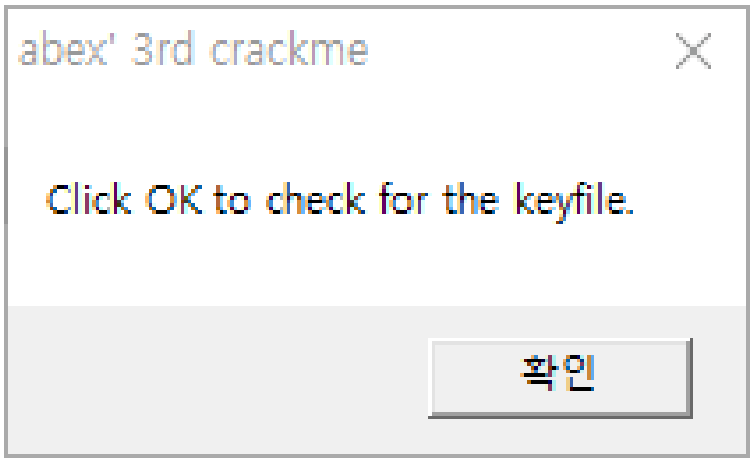
구문이 잘 출력 되는 모습이다.

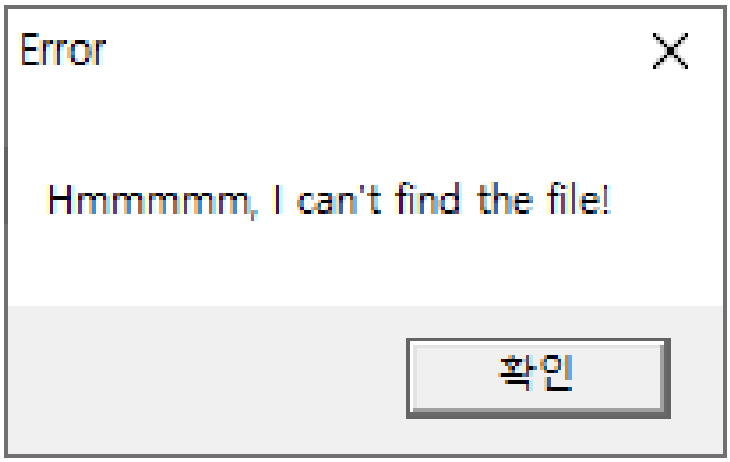
여기서 좀 헷갈릴 수 있는데, 코드엔진 페이지에 키를 입력 할 때, 바이너리 코드로 된

"6A00680020400068122040000" 이 부분을 입력 해주어야 한다.

정답은 "**6A00680020400068122040000**"

물론 key는 알아 냈지만 프로그램이 원하는 부분을 충족시켜주지 않았으니
패치한 파일로 충족시켜주도록 하자.

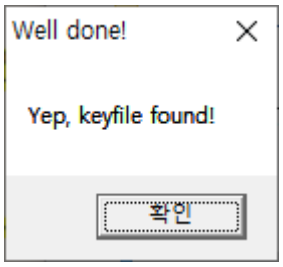




파일을 찾을 수 없다고 뜨는 것을 보니 파일을 만들어주어야 할 것 같다.

```
00401013    6A 00           push 0
00401015    68 80000000     push 80
0040101A    6A 03           push 3
0040101C    6A 00           push 0
0040101E    6A 00           push 0
00401020    68 00000080     push 80000000
00401025    68 B9204000     push patch.4020B9                ; 4020B9:"abex.l2c"
0040102A    E8 5E000000     call <patch.sub_40108D>
0040102F    A3 CA204000     mov dword ptr ds:[4020CA],eax
00401034    83F8 FF         cmp eax,FFFFFFFF
00401037    74 3C           je patch.401075
00401039    6A 00           push 0
0040103B    FF35 CA204000   push dword ptr ds:[4020CA]
00401041    E8 4D000000     call <JMP.&GetFileSize>
00401046    83F8 12         cmp eax,12
00401049    75 15           jne patch.401060
0040104B    6A 00           push 0
0040104D    68 35204000     push patch.402035                ; 402035:"Well done!"
00401052    68 40204000     push patch.402040                ; 402040:"Yep, keyfile found!"
00401057    6A 00           push 0
00401059    E8 41000000     call <JMP.&MessageBoxA>
0040105E    EB 28           jmp patch.401088
00401060    6A 00           push 0
00401062    68 79204000     push patch.402079                ; 402079:"Error"
00401067    68 7F204000     push patch.40207F                ; 40207F:"The found file is not a valid keyfile!"
0040106C    6A 00           push 0
0040106E    E8 2C000000     call <JMP.&MessageBoxA>
00401073    EB 13           jmp patch.401088
00401075    6A 00           push 0
00401077    68 54204000     push patch.402054                ; 402054:"Error"
0040107C    68 5A204000     push patch.40205A                ; 40205A:&L"te.io"
```

"abex.l2c" 라는 파일을 읽어 와서 파일의 크기가 0x12(18)byte이면 well done이 출력 될 것 같다.
18바이트 크기를 가진 파일을 만들어주고 실행시켜보면, 나는 a를 18개 입력한 txt 파일을 확장자를 다르게 바꾸었다.



성공!