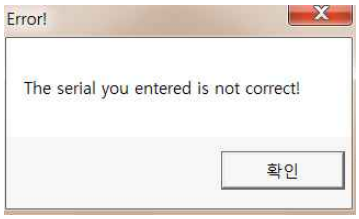
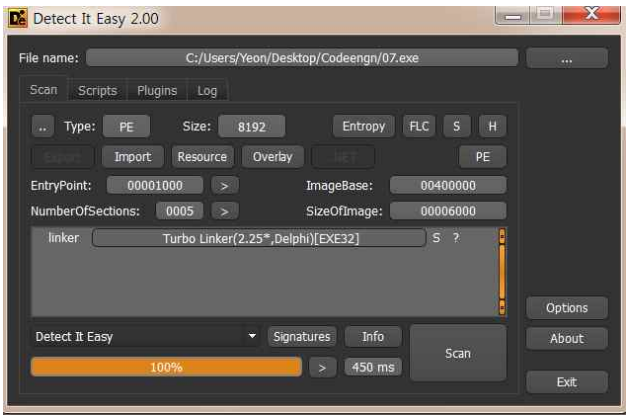
코드 엔진 Challenges: Basic 07

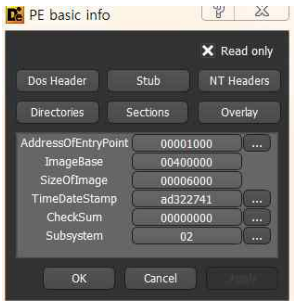
Author: abex
Korean: 컴퓨터 C드라이브의 이름이 CodeEngn 일 경우 시리얼이 생성될 때 CodeEngn 은 "어떤 것"으로 변경되는가

문제를 확인했으니 파일을 다운로드 받아서 실행해보자.



실행해보니 보통의 시리얼 인증프로그램과 유사하다. 임의의 값을 넣어봤으나 역시나 에러를 표시한다.
파일을 올리디버거로 분석하기 전에 DE를 이용해 특이점이 있나 살펴보자 .



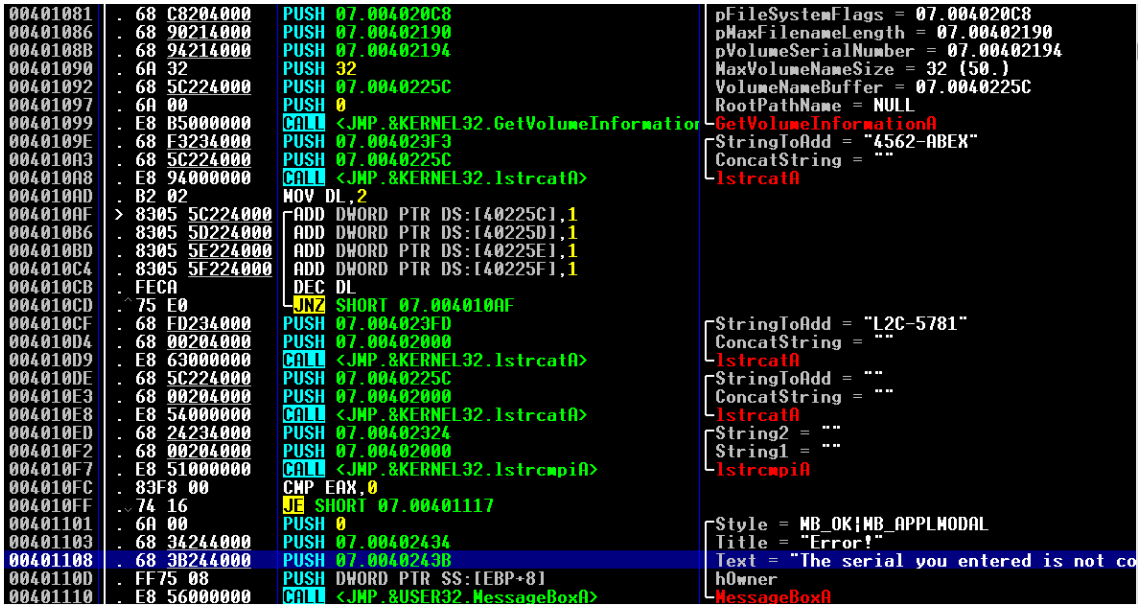


패킹도 되어 있지않고 따로 특이한 점은 없는 것 같다. Entrypoint는 00401000이다.

본격적으로 분석을 해보자. 아까 본 오류 문구인 "The serial you ~"문구를 찾아 해당 코드 영역으로 이동해보자.

```
00401000 PUSH 0                          (Initial CPU selection)
0040109E PUSH 07.004023F3                ASCII "4562-ABEX"
004010CF PUSH 07.004023FD                ASCII "L2C-5781"
00401103 PUSH 07.00402434                ASCII "Error!"
00401108 PUSH 07.0040243B                ASCII "The serial you entered is not correct!"
00401119 PUSH 07.00402406                ASCII "Well Done!"
0040111E PUSH 07.00402411                ASCII "Yep, you entered a correct serial!"
```

그 이전에 "4562-ABEX"와 "L2C-5781"이 수상하다는 것을 짚고넘어가자.



Error문구 위에 아까 수상했던 시리얼로 보이는 문자열이 있다. 이 부분을 분석해보자.

모르는 함수가 많이 나온다.
함수를 먼저 정리해보도록 하자.

#GetDlgItemText
//대화 상자에서 컨트롤과 연결된 제목 또는 텍스트를 검색합니다.
GetDlgItemText(
    _In_ HWND hDlg, //컨트롤이 들어있는 대화 상자의 핸들이다.
    _In_ int nIDDlgItem, //제목 또는 텍스트를 검색 할 컨트롤의 식별자이다.
    out LPTSTR lpString, //입력받은 텍스트를 저장하는 부분
    _In_ int nMaxCount //lpstring이 가리키는 버퍼에 복사 할 문자열의 최대 길이다.
);
반환값은 UINT로 함수가 성공하면 반환 값은 버퍼에 복사 된 문자 수를 지정하며 종료하고 NULL문자는 포함하지 않는다.
함수가 실패하면 반환값은 0이다.

파일에서는

0040106C PUSH 25	Count=25(37.)	nMaxCount
0040106E PUSH 07.00402324	Buffer=07.00402324	lpString
00401073 PUSH 63	ControlID=68(104.)	nIDDlgItem
00401075 PUSH DWORD PTR SS:[EBP+8] hWND		hDlg

00401078 CALL <JML.&USER32.GetDlgItemTextA> GetDlgItemTextA GetDlgItemTextA
대화상자에서 텍스트에 관한 정보를 얻는 함수인 GetDlgItemTextA를 이용해서 사용자가 입력한 시리얼에 대한 정보를 얻고 있다. 우리가 시리얼칸에 입력한 값은 07.00402324주소에 저장되는 것을 알 수 있다.

#BOOL WINAPI GetVolumeInformationA
//지정된 루트 디렉토리와 연관된 파일 시스템 및 볼륨에 대한 정보를 검색한다.
BOOL GetVolumeInformationA(
    _In_opt_ LPCSTR lpRootPathName,
    _Out_opt_ LPSTR lpVolumeNameBuffer,
    _In_ DWORD nVolumeNameSize,
    _Out_opt_ LPDWORD lpVolumeSerialNumber,
    _Out_opt_ LPDWORD lpMaximumComponentLength,
    _Out_opt_ LPDWORD lpFileSystemFlags,
    _Out_opt_ LPSTR lpFileSystemNameBuffer,
    _In_ DWORD nFileSystemNameSize
);
-lpRootPathName
설명 될 볼륨의 루트 디렉토리를 포함하는 문자열에 대한 포인터.
이 매개 변수가 NULL 이면 현재 디렉토리의 루트가 사용되고 뒤에 백 슬래시가 필요하다. 예를 들어 ₩ MyServer ₩ MyShare를 "₩ MyServer ₩ MyShare"로 지정하거나 C 드라이브를 "C :"로 지정한다.
-lpVolumeNameBuffer
지정된 볼륨의 이름을 수신하는 버퍼에 대한 포인터. 버퍼 크기는 nVolumeNameSize 매개 변수에 의해 지정된다.
-nVolumeNameSize
볼륨 이름 버퍼의 길이 (TCHAR) . 최대 버퍼 크기는 MAX_PATH +1이고, 볼륨 이름 버퍼가 제공되지 않으면이 매개 변수는 무시된다.
-lpVolumeSerialNumber
볼륨 일련 번호를 수신하는 변수에 대한 포인터. 일련 번호가 필요하지 않으면 이 매개 변수는 NULL 일 수 있다.
*이 함수는 하드 디스크가 포맷 될 때 운영 체제가 할당하는 볼륨 일련 번호를 반환한다.
-lpMaximumComponentLength
지정된 파일 시스템이 지원하는 파일 이름 구성 요소 의 최대 길이 (TCHAR)를받는 변수에 대한 포인터 .
파일 이름 구성 요소는 백 슬래시 사이의 파일 이름 부분이다.
* lpMaximumComponentLength가 가리키는 변수에 저장된 값은 지정된 파일 시스템이 긴 이름을 지원함을 나타내는 데 사용된다. 예를 들어 긴 이름을 지원하는 FAT 파일 시스템의 경우이 함수는 이전 8.3 표시기가 아닌 255 값을 저장한다. 긴 이름은 NTFS 파일 시스템을 사용하는 시스템에서도 지원 될 수 있다.
-lpFileSystemFlags
지정된 파일 시스템과 연관된 플래그를 수신하는 변수에 대한 포인터.
이 매개 변수는 다음 플래그 중 하나 이상일 수 있다. 그러나 FILE_FILE_COMPRESSION 및 FILE_VOL_IS_COMPRESSED 는 상호 배타적이다.
-lpFileSystemNameBuffer

파일 시스템의 이름 (예 : FAT 파일 시스템 또는 NTFS 파일 시스템)을 수신하는 버퍼에 대한 포인터로 버퍼 크기는 nFileSystemNameSize 매개 변수에 의해 지정된다 .
-nFileSystemNameSize
파일 시스템 이름 버퍼의 길이 (TCHAR) . 최대 버퍼 크기는 MAX_PATH +1이며,
파일 시스템 이름 버퍼가 제공되지 않으면 이 매개 변수는 무시된다.
-반환 값
요청 된 모든 정보가 검색되면 반환 값은 0이 아닙니다.
요청 된 정보가 모두 검색되지 않으면 반환 값은 0입니다. 확장 된 오류 정보를 얻으려면 GetLastError를 호출하십시오 .

지정된 루트 디렉터리에 관한 시스템 볼륨 정보를 얻는 함수인 GetVolumeInformation을 이용해서 볼륨정보를 얻고 있다. 다양한 정보가 입력되지만 우리가 주목할 부분은 드라이브의 이름에 대한 부분이다. 여기서 VolumeNameBuffer에 드라이브의 이름이 저장된다.

```
0040107D        PUSH 0          pFileSystemNameSize = NULL     nFileSystemNameSize
0040107F        PUSH 0          pFileSystemNameBuffer = NULL lpFileSystemNameBuffer
00401081        PUSH 07.004020C8 pFileSystemFlags = 07.004020C8 lpFileSystemFlags
00401086        PUSH 07.00402190 pMaxFilenameLength = 07.00402190
lpMaximumComponentLength
0040108B        PUSH 07.00402194 pVolumeSerialNumber = 07.00402194
lpVolumeSerialNumber
00401090        PUSH 32         MaxVolumeNameSize = 32 (50.) nVolumeNameSize
00401092        PUSH 07.0040225C VolumeNameBuffer = 07.0040225C
lpVolumeNameBuffer
00401097        PUSH 0          RootPathName = NULL   lpRootPathName
00401099        CALL 00401153   GetVolumeInformationA   GetVolumeInformation
```

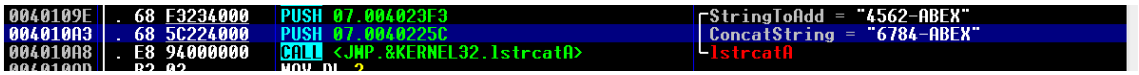
우리가 찾는 드라이브의 이름은 004010225C 주소에 저장되는 것을 알 수 있다. 이를 이용해서 드라이브 이름을 변경할 수 있다.

```
#lstrcat
//한 문자열을 다른 문자열에 추가합니다.
LPSTR lstrcatA(
 _Inout_ LPSTR   lpString1,   //ConcatString
 _In_ LPCSTR lpString2        //StringToAdd
);
```

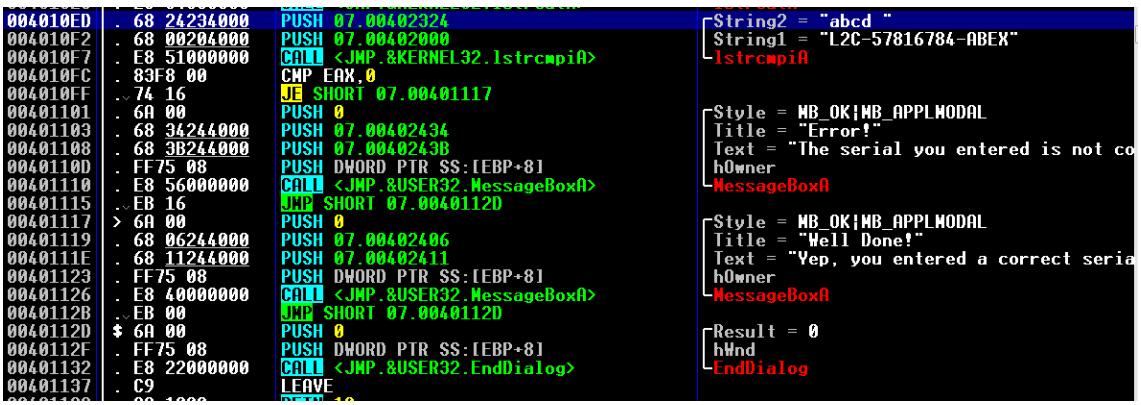
-lpString1
첫 번째 널 종료 문자열. 이 버퍼는 두 문자열을 모두 포함 할 수있을만큼 커야한다.
-lpString2
lpString1 매개 변수에 지정된 문자열에 추가 될 Null 종료 문자열 .
-반환 값(유형 : LPTSTR)
함수가 성공하면 반환 값은 버퍼에 대한 포인터로 함수가 실패하면 리턴 값은 NULL 이고 lpString1 은 널 (NULL) 종료되지 않을 수 있다.

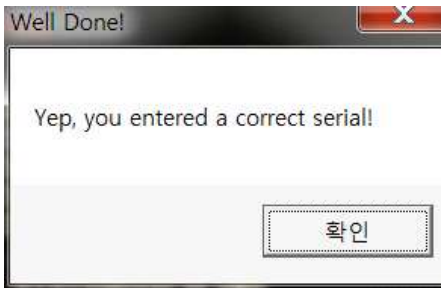
lpString1(ConcatString) + lpString2(StringToAdd) 값을 반환하는 함수이다.
0040109E PUSH 07.004023F3 StringToAdd = "4562-ABEX" lpString2
004010A3 PUSH 07.0040225C (디스크 이름) ConcatString = "" lpString1
004010A8 CALL <JMP.&KERNEL32.lstrcatA> lstrcatA lstrcat

위의 같은 경우는 "" + "4562-ABEX" 값인 "4562-ABEX"를 반환하여 0040225C 에 저장한다.



004010ED~004010F7(lstrcmpiA)
00402324(입력값)과 00402000 위의 과정을 거쳐서 얻은 시리얼 값(L2C-57816784-ABEX)를 lstrcmpiA함수를 이용해서 비교 하고 있다.
*lstrcmpiA함수sms 비교 값이 같으면 0을 반환하므로 JE문을 이용해서 메시지를 띄우게 된다.



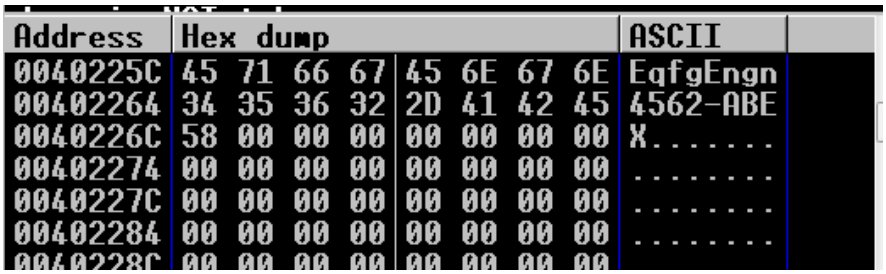


시리얼 값은 L2C-57816784-ABEX이다.

문제의 답을 풀기 위해서 C:드라이브의 이름을 CodeEngn으로 바꾸고 프로그램을 다시 실행해보았다.

```
00401092  . 68 5C224000    PUSH 07.0040225C                 VolumeNameBuffer = 07.0040225C
00401097  . 6A 00           PUSH 0                           RootPathName = NULL
00401099  . E8 B5000000     CALL <JMP.&KERNEL32.GetVolumeInformatio  GetVolumeInformationA
0040109E  . 68 E3234000     PUSH 07.004023F3                 StringToAdd = "4562-ABEX"
004010A3  . 68 5C224000     PUSH 07.0040225C                 ConcatString = "EqfgEngn4562-ABEX"
004010A8  . E8 94000000     CALL <JMP.&KERNEL32.lstrcatA>    lstrcatA
004010AD  . B2 02           MOV DL,2
004010AF  > 8305 5C224000   ADD DWORD PTR DS:[40225C],1
004010B6  . 8305 5D224000   ADD DWORD PTR DS:[40225D],1
004010BD  . 8305 5E224000   ADD DWORD PTR DS:[40225E],1
004010C4  . 8305 5F224000   ADD DWORD PTR DS:[40225F],1
004010CB  . FECA            DEC DL
004010CD  . 75 E0           JNZ SHORT 07.004010AF
004010CF  . 68 FD234000     PUSH 07.004023FD                 StringToAdd = "L2C-5781"
004010D4  . 68 00204000     PUSH 07.00402000                 ConcatString = ""
004010D9  . E8 63000000     CALL <JMP.&KERNEL32.lstrcatA>    lstrcatA
004010DE  . 68 5C224000     PUSH 07.0040225C                 StringToAdd = "EqfgEngn4562-ABEX"
004010E3  . 68 00204000     PUSH 07.00402000                 ConcatString = ""
004010E8  . E8 54000000     CALL <JMP.&KERNEL32.lstrcatA>    lstrcatA
004010ED  . 68 24234000     PUSH 07.00402324                 String2 = "Enter your serial"
004010F2  . 68 00204000     PUSH 07.00402000                 String1 = ""
004010F7  . E8 51000000     CALL <JMP.&KERNEL32.lstrcmpiA>   lstrcmpiA
004010FC  . 83F8 00         CMP EAX,0
```

GetVolumeImformation을 통해 VolumeNameBuffer인 0040225C에 CodeEngn이라는 드라이브의 이름을받아오고 lstcatA 함수를 이용해 두 개의 문자열을 합친다. 그 후 004010AF~004010CD를 연산하면서 EqfqEngn4562-ABEX를 만든다. 메모리 할당 값은 아래와 같다.

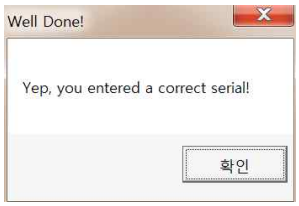


Address	Hex dump								ASCII
0040225C	45	71	66	67	45	6E	67	6E	EqfgEngn
00402264	34	35	36	32	2D	41	42	45	4562-ABE
0040226C	58	00	00	00	00	00	00	00	X.......
00402274	00	00	00	00	00	00	00	00	....
0040227C	00	00	00	00	00	00	00	00	....
00402284	00	00	00	00	00	00	00	00	....
0040228C	00	00	00	00	00	00	00	00	....

CodeEngn가 Eqfg로 바뀐 후 코드를 실행해보면 L2C-5781과 EqfgEngn4562-ABEX와 합쳐친 뒤 String1이 되며 비교 값이 같으면 0을 반환하는 lstrcmpiA함수를 통해 string1과 사용자 임의의 시리얼 입력 값인 "abcd"와 비교하면 두 값이 다르기에 리턴 값이 1이기에 Error 가 나온다. 그렇다면 abcd대신 L2C-5781을 넣어 L2C-5781EqfgEngn4562-ABEX 입력해 확인해 보자.

```
004010CD  . 75 E0           JNZ SHORT 07.004010AF
004010CF  . 68 FD234000     PUSH 07.004023FD                 StringToAdd = "L2C-5781"
004010D4  . 68 00204000     PUSH 07.00402000                 ConcatString = "L2C-5781EqfgEngn4562-AB
004010D9  . E8 63000000     CALL <JMP.&KERNEL32.lstrcatA>    lstrcatA
004010DE  . 68 5C224000     PUSH 07.0040225C                 StringToAdd = "EqfgEngn4562-ABEX"
004010E3  . 68 00204000     PUSH 07.00402000                 ConcatString = "L2C-5781EqfgEngn4562-AB
004010E8  . E8 54000000     CALL <JMP.&KERNEL32.lstrcatA>    lstrcatA
004010ED  . 68 24234000     PUSH 07.00402324                 String2 = "L2C-5781EqfgEngn4562-ABEX"
004010F2  . 68 00204000     PUSH 07.00402000                 String1 = "L2C-5781EqfgEngn4562-ABEX"
004010F7  . E8 51000000     CALL <JMP.&KERNEL32.lstrcmpiA>   lstrcmpiA
004010FC  . 83F8 00         CMP EAX,0
004010FF  . 74 16           JE SHORT 07.00401117
00401117  > 6A 00           PUSH 0                           Style = MB_OK|MB_APPLMODAL
00401119  . 68 06244000     PUSH 07.00402406                 Title = "Well Done!"
0040111E  . 68 11244000     PUSH 07.00402411                 Text = "Yep, you entered a correct seri
00401123  . FF75 08         PUSH DWORD PTR SS:[EBP+8]        hOwner
00401126  . E8 40000000     CALL <JMP.&USER32.MessageBoxA>   MessageBoxA
```

두 입력값이 같아 EAX에 0이 반환되므로 두 값이 같아 0040117로 점프해 확인 메시지가 나온다.



컴퓨터 C드라이브의 이름이 CodeEngn일 경우 시리얼이 생성값은 L2C-5781EqfgEngn4562-ABEX이 것이며 CodeEngn은 EqfgEngn으로 바뀐다.