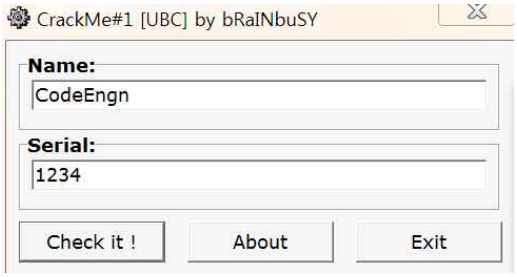
코드 엔진 Challenges: Basic 15

Author: uBc-bRiANbuSY
Korean: Name이 CodeEngn을 구하시오

문제를 보니 14번 문제와 같이 시리얼 생성 관련 알고리즘을 파악하는 문제같다. 이 문제 또한 이름 값에 따라서 시리얼 값이 변할 것이라는 것을 예측할 수 있다. 문제를 파악했으니 프로그램을 실행해보자.



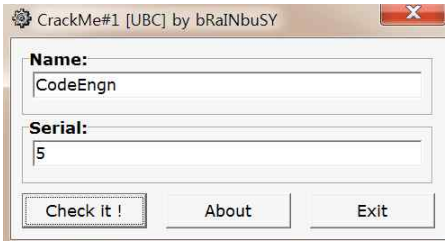
코드를 분석하기 전에 DE를 통해 패킹 여부를 살펴보니 패킹은 되어있지 않다. 올리디버거를 통해 분석해보도록 하자. 먼저 문자열 검색을 통해 아까 본 메시지가 실행되는 주소로 이동하자. 성공시 출력되는 메시지와 실패시 출력되는 메시지가 비슷한 주소에 있는 것을 보아 분기점이 있을 것이라는 것을 예측할 수 있다.

```
00458757 ASCII "Unit1"
00458839 MOV EAX,15.00458888                ASCII "You cracked the UBC CrackMe#1 ! Please send your solution to ubcrackers@hotmail.com
00458843 MOV EDX,15.004588E8                ASCII "CRACKED"
00458854 MOV EAX,15.004588F8                ASCII "Try Again !"
00458888 ASCII "You cracked the "
00458898 ASCII "UBC CrackMe#1 ! "
004588A8 ASCII "Please send your"
004588B8 ASCII " solution to ubc"
004588C8 ASCII "rackers@hotmail."
004588D8 ASCII "com !",0
004588E8 ASCII "CRACKED",0
004588F8 ASCII "Try Again !",0
00458AB0 PUSH EBP                           (Initial CPU selection)
00458AD3 MOV EDX,15.00458B28                ASCII "CrackMe#1 [UBC] by bRaINbuSY"
00458B28 ASCII "CrackMe#1 [UBC] "
00458B38 ASCII "by bRaINbuSY",0
```

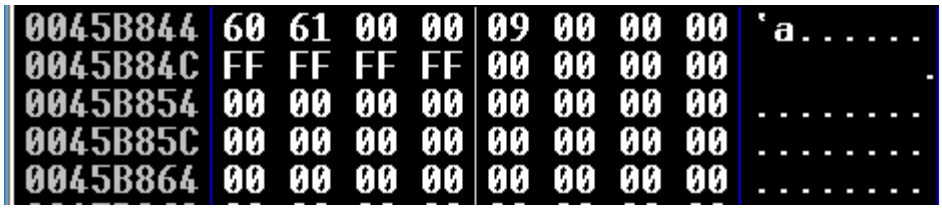
```
0045881E  . 8B83 D0020000  MOV EAX,DWORD PTR DS:[EBX+2D0]
00458824  . E8 97CDFCFF    CALL 15.004255C0
00458829  . 8B45 FC        MOV EAX,DWORD PTR SS:[EBP-4]
0045882C  . E8 43EFFAFF    CALL 15.00407774
00458831  . 3B05 44B84500  CMP EAX,DWORD PTR DS:[45B844]
00458837  .v75 1B          JNZ SHORT 15.00458854
00458839  . B8 88884500    MOV EAX,15.00458888              ASCII "You cracked the
0045883E  . E8 29C1FEFF    CALL 15.0044496C
00458843  . BA E8884500    MOV EDX,15.004588E8              ASCII "CRACKED"
00458848  . A1 3CB84500    MOV EAX,DWORD PTR DS:[45B83C]
0045884D  . E8 9ECDFCFF    CALL 15.004255F0
00458852  .vEB 0A          JMP SHORT 15.0045885E
00458854  > B8 F8884500    MOV EAX,15.004588F8              ASCII "Try Again !"
00458859  . E8 0EC1FEFF    CALL 15.0044496C
0045885E  > 33C0           XOR EAX,EAX
00458860  . 5A             POP EDX
00458861  . 59             POP ECX
00458862  . 59             POP ECX
00458863  . 64:8910        MOV DWORD PTR FS:[EAX],EDX
00458866  . 68 7B884500    PUSH 15.0045887B
0045886B  > 8D45 FC        LEA EAX,DWORD PTR SS:[EBP-4]
```

역시 분기문이 있는 것을 확인 할 수 있다. 또한 14번 문제와 비슷하게 시리얼 값을 비교하

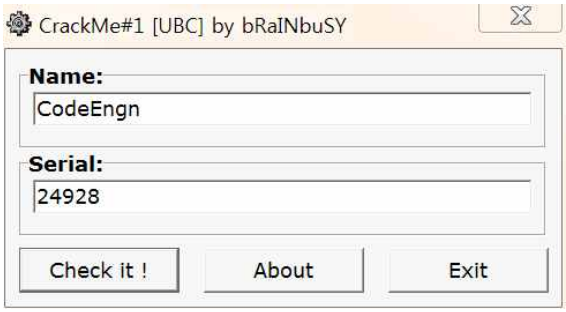
는 듯한 CMP문 또한 확인 가능하다. 코드를 살펴보니 EAX값과 DS값(45B844주소의 4Byte) 값을 비교해서 다르면 00458854(Try Again!)으로 점프하고 같으면 성공메시지를 출력하는 것을 알 수있다. 비교하는 2개의 값을 알았으니 분기문에 BP를 설정하고 두 개중 어떤ㄴ 것이 시리얼 값이 저장된 것인지 확인하여 시리얼값을 찾아보자.

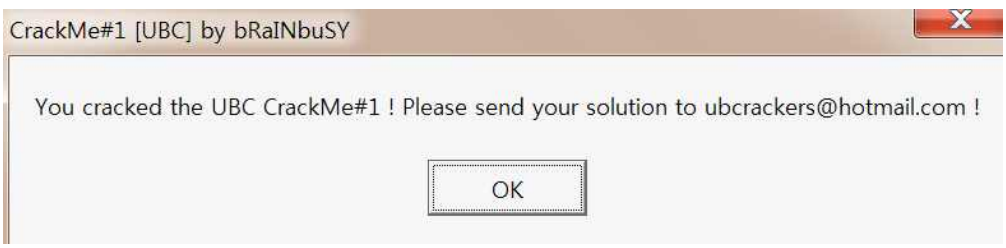


일단 EAX값을 보면 5가 저장되어있는 것을 확인할 수 있다. 입력한 시리얼값이 5라는 것을 생각했을 때 사용자가 입력한 값은 EAX에 저장되는 것을 확인할 수 있다.
이제 시리얼이 45B884주소에 있다는 것을 확인했으니 Follow in Dump를 이용해서 주소로 이동해보자.



45B844의 4byte에는 00006160이 저장된 것이 보인다. 이를 10진수로 변환한 값인 24928이 시리얼 값이라는 것을 알 수 있다 .확인을 위해 프로그램을 돌려 인증해보자.





인증에 성공한 것을 확인할 수 있다.