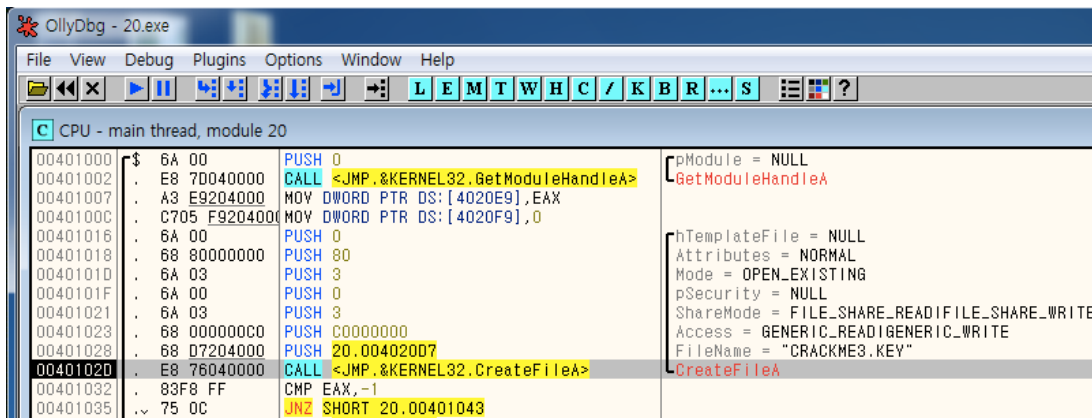
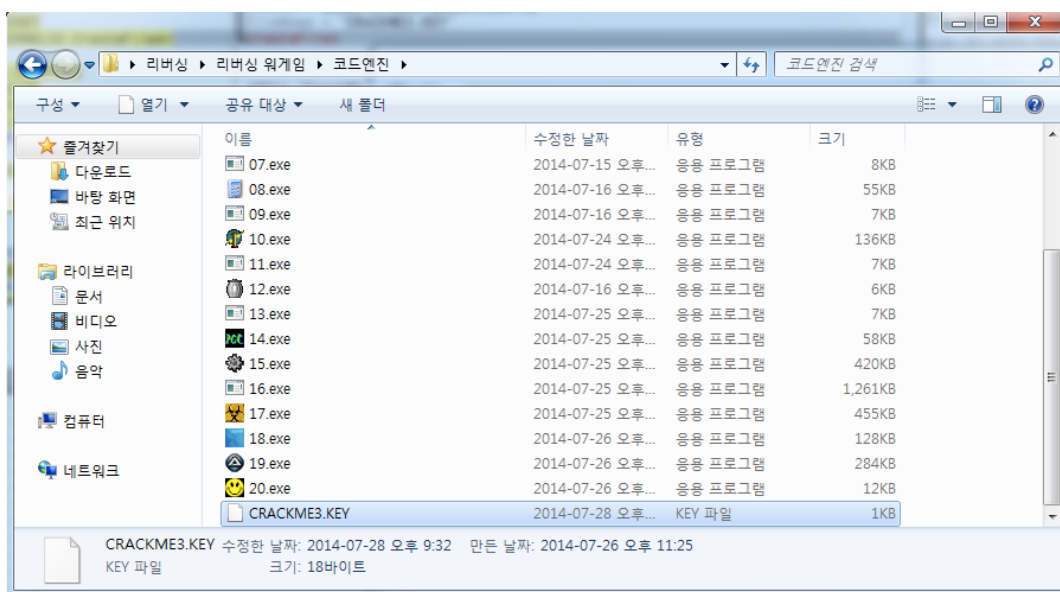


처음 실행화면입니다. 아무런 반응이 보이지 않습니다.



올리디버거를 통해 분석해 보니 CRACKME3.KEY가 없으면 반응을 하지 않는것을 알 수 있습니다.

CreateFile 함수를 통해 해당 파일이 있는지 없는지 체크하게 됩니다.



파일을 만들었습니다.

```

CPU - main thread, module 20
00401052 . 6A 00          PUSH 0
00401054 . 68 A0214000    PUSH 20.004021A0
00401059 . 50             PUSH EAX
0040105A . 53             PUSH EBX
0040105B . FF35 F5204000 PUSH DWORD PTR DS:[4020F5]
00401061 . E8 30040000    CALL <JMP.&KERNEL32.ReadFile>

```

p0verlapped = NULL  
 pBytesRead = 20.004021A0  
 ByteToRead = 12 (18.)  
 Buffer => 20.00402008  
 hFile = 000000EC (window)  
 ReadFile

ReadFile을 통해 18byte를 읽는 것을 알 수 있습니다.

만일 파일이 18byte에 미치지 못하면 역시 반응하지 않습니다.

그래서 파일에 123456789123456789 를 넣고 다시 진행합니다.

```

CPU - main thread, module 20
0040106D . ^ 75 C8          JNZ SHORT 20.00401037
0040106F . 68 08204000    PUSH 20.00402008
00401074 . E8 98020000    CALL 20.00401311
00401079 . 8135 F9204000  XOR DWORD PTR DS:[4020F9],12345678
00401083 . 83C4 04        ADD ESP,4
00401086 . 68 08204000    PUSH 20.00402008
0040108B . E8 AC020000    CALL 20.0040133C
00401090 . 83C4 04        ADD ESP,4
00401093 . 3B05 F9204000  CMP EAX,DWORD PTR DS:[4020F9]
00401099 . 0F94C0         SETE AL
0040109C . 50             PUSH EAX
0040109D . 84C0           TEST AL,AL
0040109F . ^ 74 96         JE SHORT 20.00401037
004010A1 . 68 0E214000    PUSH 20.0040210E
004010A6 . E8 98020000    CALL 20.00401346
004010AB . 83C4 04        ADD ESP,4
004010AE . > 6A 00          PUSH 0
004010B0 . 68 28214000    PUSH 20.00402128
004010B5 . E8 9A030000    CALL <JMP.&USER32.FindWindowA>
004010BA . 0BC0           OR EAX,EAX
004010BC . ~ 74 01        JE SHORT 20.004010BF
004010BE . C3             RETN

```

ASCII "123456789123456789"  
 알고리즘  
 ASCII "123456789123456789"  
 ASCII "CrackMe v3.0"  
 Title = NULL  
 Class = "No need to disasm the code!"  
 FindWindowA

지금 버퍼에 파일의 값이 들어가있고 알고리즘 이라고 주석이 되어있는 부분이 있습니다.

한번 분석해보도록 하겠습니다.

```

CPU - main thread, module 20
00401311 $ 33C9 XOR ECX,ECX
00401313 . 33C0 XOR EAX,EAX
00401315 . 8B7424 04 MOV ESI,DWORD PTR SS:[ESP+4]
00401319 . B3 41 MOV BL,41
0040131B > 8A06 MOV AL,BYTE PTR DS:[ESI]
0040131D . 32C3 XOR AL,BL
0040131F . 8806 MOV BYTE PTR DS:[ESI],AL
00401321 . 46 INC ESI
00401322 . FEC3 INC BL
00401324 . 0105 F9204000 ADD DWORD PTR DS:[4020F9],EAX
0040132A . 3C 00 CMP AL,0
0040132C . 74 07 JE SHORT 20.00401335
0040132E . FEC1 INC CL
00401330 . 80FB 4F CMP BL,4F
00401333 . 75 E6 JNZ SHORT 20.0040131B
00401335 > 890D 49214000 MOV DWORD PTR DS:[402149],ECX
0040133B . C3 RETN
0040133C $ 8B7424 04 MOV ESI,DWORD PTR SS:[ESP+4]
00401340 . 83C6 0E ADD ESI,0E
00401343 . 8B06 MOV EAX,DWORD PTR DS:[ESI]
00401345 . C3 RETN
00401346 $ 8B7424 04 MOV ESI,DWORD PTR SS:[ESP+4]
0040134A . 83C6 00 ADD ESI,00
0040134D . C706 20202043 MOV DWORD PTR DS:[ESI],43202020
00401353 . C746 04 72616 MOV DWORD PTR DS:[ESI+4],6B636172
0040135A . C746 08 65642 MOV DWORD PTR DS:[ESI+8],21216465
00401361 . C3 RETN

Local call from <ModuleEntryPoint>+74

Address Hex dump ASCII
00402000 00 00 00 00 00 00 00 00 .....
00402008 70 70 70 70 70 70 70 70 pppppppp
00402010 70 7B 79 7F 79 7B 36 37 p{y0y{67
00402018 38 39 00 00 00 00 00 00 89.....

```

해당 함수 입니다. ESI에는 입력받은 문자열의 전체가 들어가고 AL에는 한글자씩이 들어갑니다.

BL에는 0x41이 들어간 것을 확인할 수 있고 AL과 BL과 XOR연산을 합니다.

그 결과는 하단의 검정네모처럼 바로 ESI안의 원래 문자열과 바뀌게 됩니다.

이런 과정을 반복하게 되는데 XOR한 결과값이 0이거나 BL이 4F이면 이 루틴을 빠져나오게 됩니다.

즉, BL이 A부터 O까지 변하는데 마지막 O는 연산을 안하니 14번의 변화가 이루어집니다.

그리고 네모를 친 부분이 이후에 인증하는데 중요한 역할을 하게 됩니다.

해당 주소는0x004020F0이고 이 주소에 XOR한 연산의 결과를 전부 더하게 됩니다.

알고리즘을 빠져나왔는데 밑줄 친 부분을 잘 보시기 바랍니다.

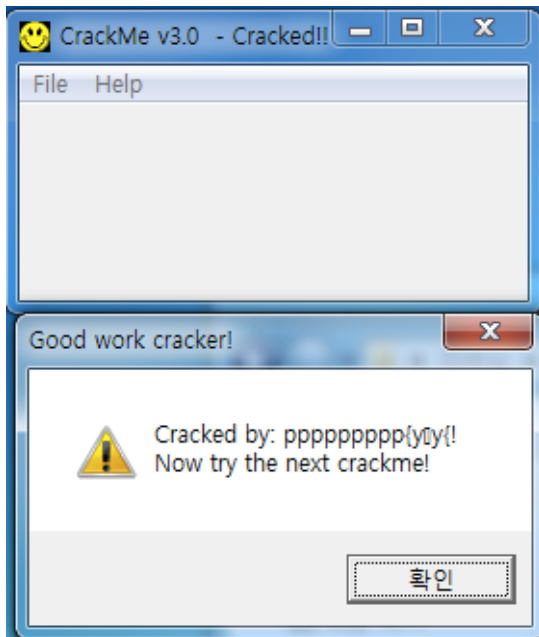
위에서 중요하다고 했던 그 주소인데 여기에는 그간 연산결과와 합이 들어있습니다.

그 결과와 0x12345678을 XOR하게 됩니다.

그 바로 밑의 부분인데 주석을 보면 '뒤에 4개만 뽑는 함수'라고 되어있습니다.

아까 전에 알고리즘에서 연산할 때 14개만 연산하고 나머지 4개는 연산하지 않았습니다.

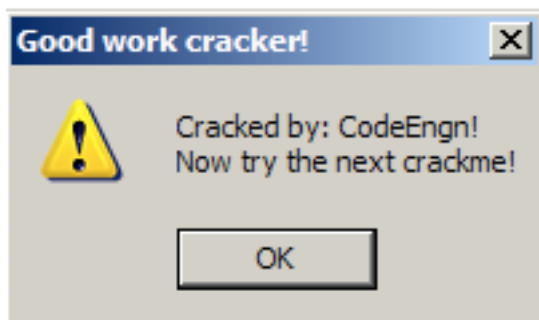
그 4개의 HEX값이 이 함수를 지나면 EAX로 들어오고 0x004020F9와 비교하게 되고 분기합니다.



그냥 조건을 다 건너뛰고 실행해본 결과 XOR한 연산 결과값이 출력되는 것을 확인할 수 있습니다.

자 그러면 여태까지 분석한 것들을 정리해 보면 아래와 같습니다.

1. 파일에서 18byte의 값을 받아온다.
2. 1번의 값 각각 한 글자씩 A~N까지 XOR한다.
3. 모든 연산의 결과는 특정 번지(ADD 번지)에 더해진다.
4. ADD 번지와 0x12345678을 XOR 연산한다.
5. 파일에서 받아온 값 중 뒤의 4byte의 HEX값과 4번의 결과와 비교한다.



원하고자 하는 바는 이러한 화면을 띄우는 것입니다.

XOR의 특성 상  $A \text{ XOR } B = C$  이면  $A \text{ XOR } C = B$  이기 때문에 간단하게 원하는 값을 얻을 수 있습니다.