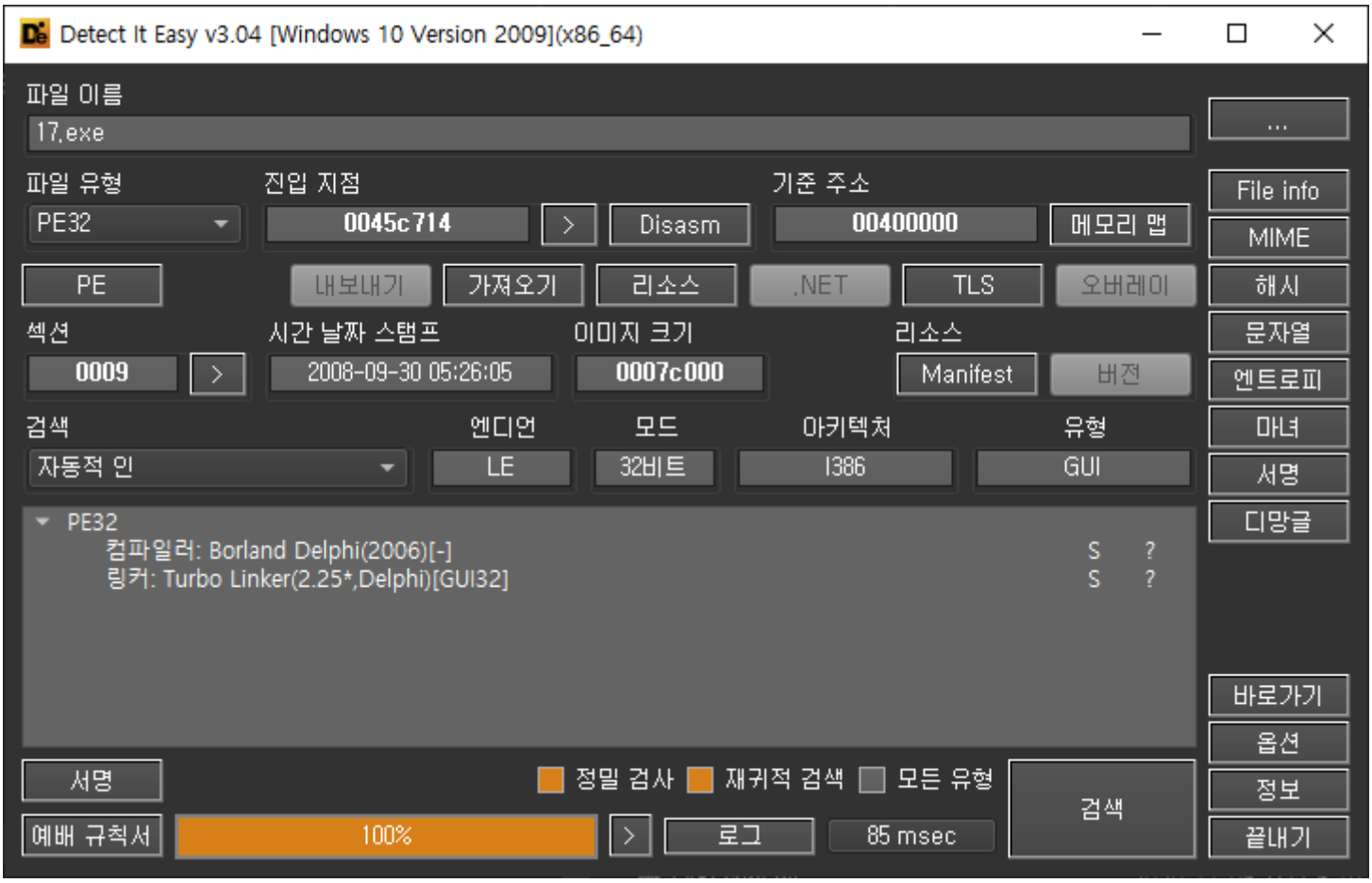
codeengn-basic-L17 풀이

리버싱 문제풀이 / Wonlf / 2022. 4. 21. 15:59

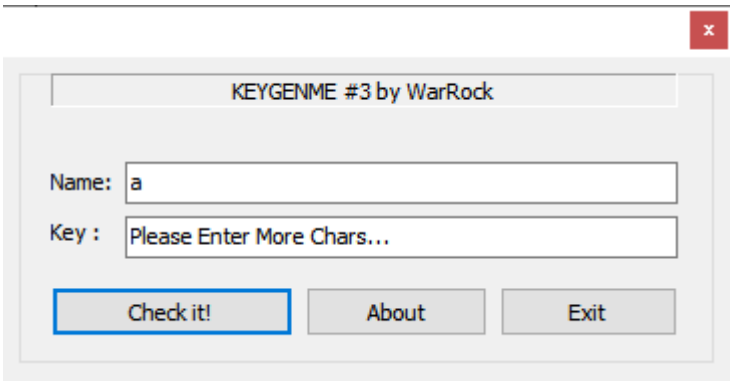
문제는 Key가 BEDA-2F56-BC4F4368-8A71-870B 일때 한글자인 name의 MD5 해쉬값을 원하고 있다.

Die로 열어보면,



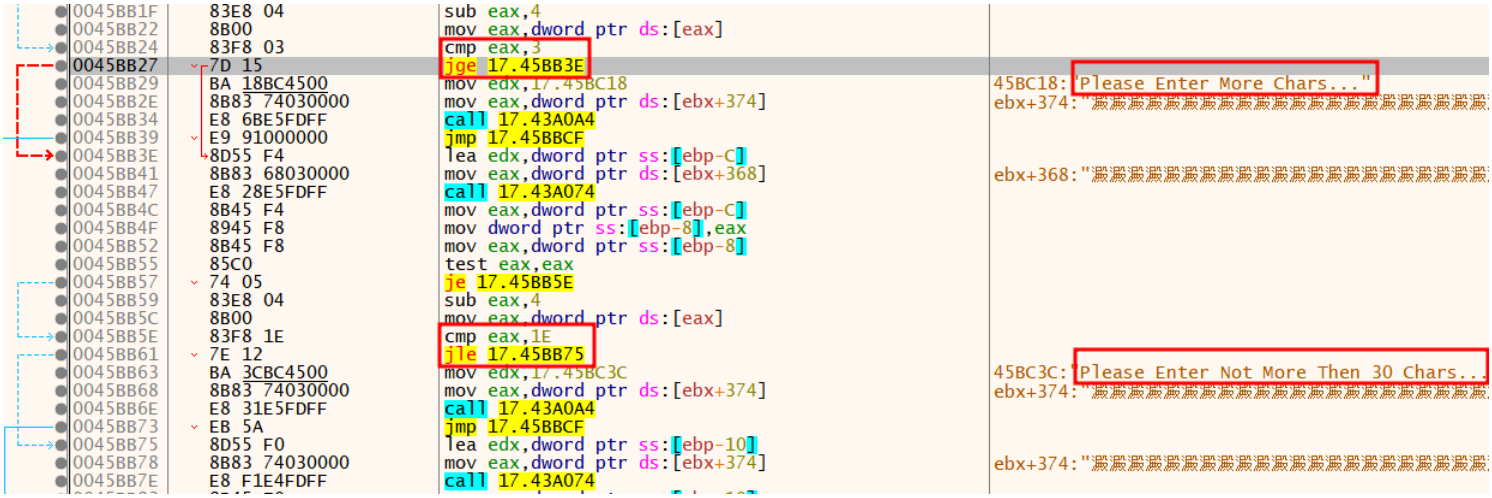
특이사항은 보이지 않는다.

파일을 실행해본다.



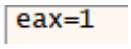
Name은 한글자라는데 한글자를 입력하면 Please Enter More Chars 더 많은 문자를 입력해달라고 한다.
이렇게 되면 Name을 구할 수 없으니 디버거로 열어서 문제를 파악해야겠다.

문자열 찾기를 사용하여 "Please Enter More Chars..." 문자열을 사용하는 곳을 찾았다.



더 아래를 보니 문자열이 30길이를 넘으면 안되는 구문도 보인다.

cmp와 점프분기가 있으니 높은 확률로 eax에는 길이가 들어갈 것 같고 실제로도 확인해보면,

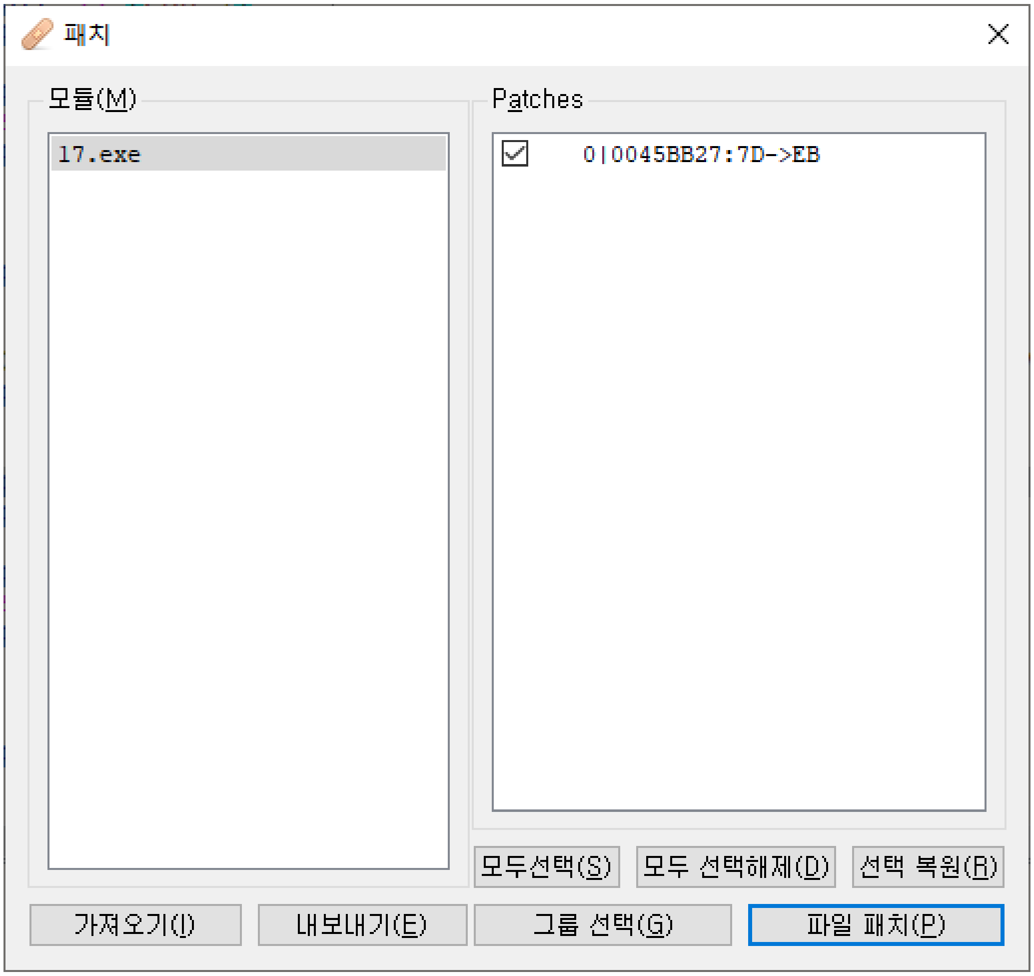


길이

길이가 맞다.

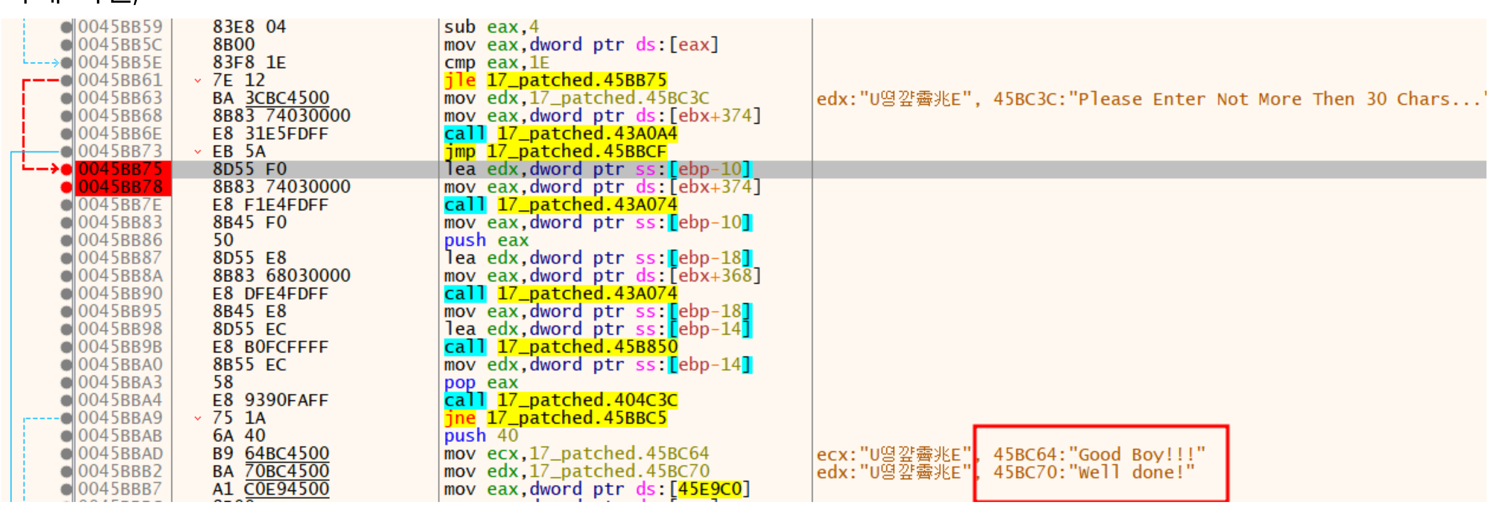
우리는 한글자만 입력하면 되니 jge쪽을 보게되면 jge는 eax >= 0x3 일때 점프를 뛴다.

점프를 뛰어야 그 다음 검증인 30길이를 넘는지 안넘는지 구문까지 갈 수 있기에 jge를 jmp로 바꾸어 주었다.

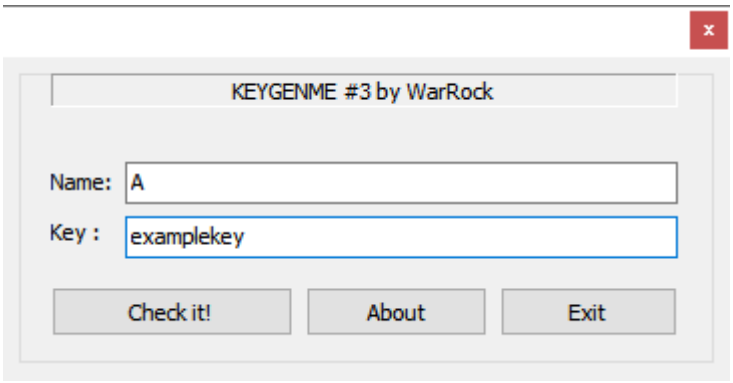


jmp로 바꾼 파일을 패치해서 저장 하고 저장한 파일을 디버깅하기 시작했다.

그리고 더 내려가면 30길이를 넘는지 안넘는지 구문은 한글자만 입력 했으니 당연히 통과 할것이고 더 아래로 가게 되면,



0045BB75부터 시리얼이 맞는지 틀린지를 판단하는 구문의 시작인 것 같아 브레이크 포인트를 걸고



특정 값을 넣은뒤 한줄씩 실행시켜보았다.

```
0045BB75   8D55 F0      lea edx,dword ptr ss:[ebp-10]      [ebp-10]:"examplekey"
0045BB78   8B83 74030000 mov eax,dword ptr ds:[ebx+374]    [ebx+374]:"뙌B"
0045BB7E   E8 F1E4FDFF  call 17_patched.43A074
0045BB83   8B45 F0      mov eax,dword ptr ss:[ebp-10]      [ebp-10]:"examplekey"
0045BB86   50           push eax
0045BB87   8D55 E8      lea edx,dword ptr ss:[ebp-18]
0045BB8A   8B83 68030000 mov eax,dword ptr ds:[ebx+368]    [ebx+368]:"뙌B"
0045BB90   E8 DFE4FDFF  call 17_patched.43A074
0045BB95   8B45 E8      mov eax,dword ptr ss:[ebp-18]
0045BB98   8D55 EC      lea edx,dword ptr ss:[ebp-14]      [ebp-14]:"FFE3-2C73-0502A34C-8A48-E1CB"
0045BB9B   E8 B0FCFFFF  call 17_patched.45B850
0045BBA0   8B55 EC      mov edx,dword ptr ss:[ebp-14]      [ebp-14]:"FFE3-2C73-0502A34C-8A48-E1CB"
0045BBA3   58           pop eax
```

ebp-10에는 Key로 입력한 "examplekey" 가 들어 있었고 더 실행하다보면,

45B850 함수를 호출 했을 때, Name에 대한 Key가 스택에 저장 되었다.

A를 넣어서 저런 Key가 나왔으니 B도 넣어서 확인해본다.

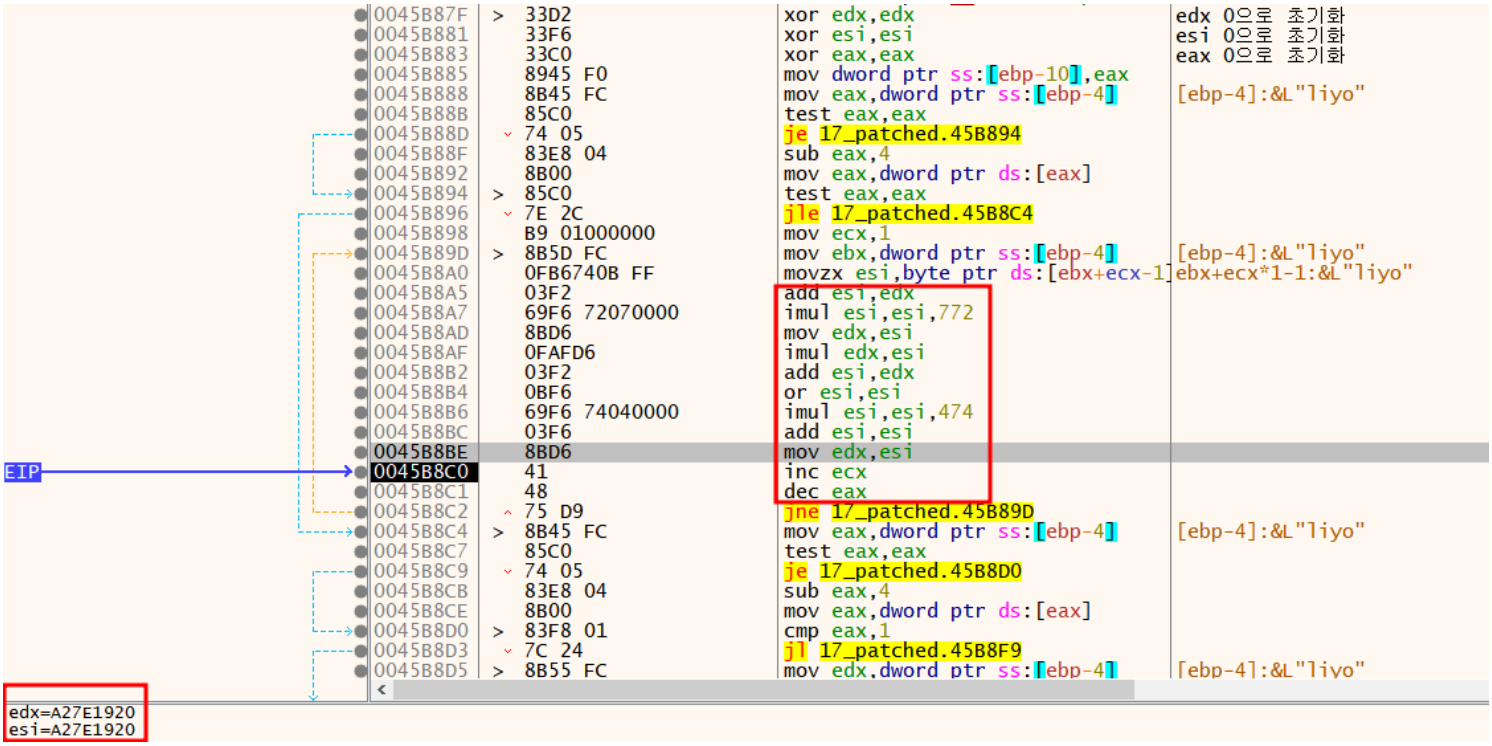
```
0045BB75   8D55 F0      lea edx,dword ptr ss:[ebp-10]      [ebp-10]:"examplekey"
0045BB78   8B83 74030000 mov eax,dword ptr ds:[ebx+374]    [ebx+374]:"뙌B"
0045BB7E   E8 F1E4FDFF  call 17_patched.43A074
0045BB83   8B45 F0      mov eax,dword ptr ss:[ebp-10]      [ebp-10]:"examplekey"
0045BB86   50           push eax
0045BB87   8D55 E8      lea edx,dword ptr ss:[ebp-18]      [ebp-18]:&L"liyo"
0045BB8A   8B83 68030000 mov eax,dword ptr ds:[ebx+368]    [ebx+368]:"뙌B"
0045BB90   E8 DFE4FDFF  call 17_patched.43A074
0045BB95   8B45 E8      mov eax,dword ptr ss:[ebp-18]      [ebp-18]:&L"liyo"
0045BB98   8D55 EC      lea edx,dword ptr ss:[ebp-14]      [ebp-14]:"A27E-2D07-BA91EB06-8A50-A5D4"
0045BB9B   E8 B0FCFFFF  call 17_patched.45B850
0045BBA0   8B55 EC      mov edx,dword ptr ss:[ebp-14]      [ebp-14]:"A27E-2D07-BA91EB06-8A50-A5D4"
0045BBA3   58           pop eax
```

이런식으로 Name에 따라서 Key가 바뀌는 것을 알 수 있는데, 여기서 사실 답을 알아버렸다.

어떠한 Name이 입력 됐을 때, key가 문제에 있던 BEDA-2F56-BC4F4368-8A71-870B 이것과 같으면 되기에 알파벳을 입력해보다 "F" 를 입력했을때 성공구문으로 가는 것을 알았다

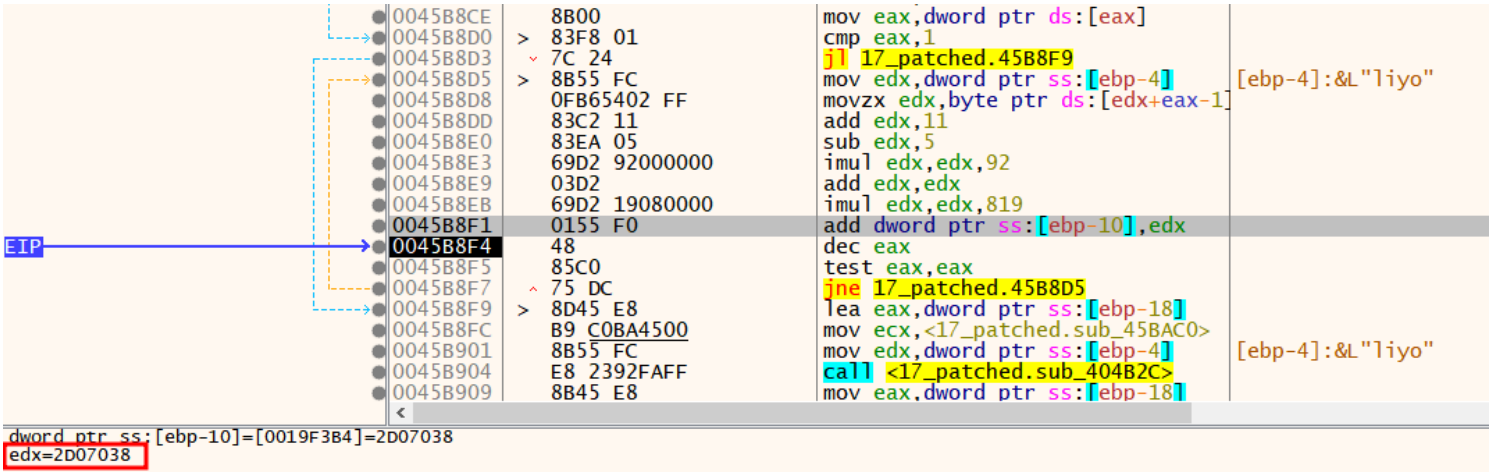
하지만 그렇게 풀면 내 자신이 한심하기 때문에 Key를 만드는 45B850 함수를 뒤져보기로 한다.

함수를 뒤져보다 보니 특정 연산을 하는 코드를 발견했다.



특정 연산을 하고 edx에 저장되는 값은 Name에 B를 입력했을 때의 Key의 앞 4자리인 A27E와 유사한 A27E1920가 들어있다.

더 아래로 진행해보면,



또 다른 특정 연산 구문이 있고 이 연산 구문을 지나고 나면 Key의 두번째인 2D07과 유사한 2D07038이 EDX에 들어있다.

이로써 추측한 바로는, 아마 더 아래로 내려 갔을때, Key의 다른 부분을 정하는 연산 코드가 있을 것이고, 나중에 그것을 조합하고 입력한 Key와 비교 할것 같다.

그리고 계속 Name에 "B"를 입력 했을때 첫번째 4자리와 두번째 4자리의 Key가 변경되지 않았으니, 또 다른 경우의 수를 찾을 필요는 없을 것 같다.

그렇기 때문에 처음 보았던 연산 코드로 Key의 첫번째 4자리만 브루트포스 해준다면 key를 구할 수 있을 것이다.

```
0045B898    B9 01000000     mov ecx,1
0045B89D  > 8B5D FC         mov ebx,dword ptr ss:[ebp-4]    [ebp-4]:&L"liyo"
0045B8A0    0FB6740B FF     movzx esi,byte ptr ds:[ebx+ecx-1]ebx+ecx*1-1:&L"liyo"
0045B8A5    03F2            add esi,edx                     edx는 0이니까 입력받은 esi = Name
0045B8A7    69F6 72070000   imul esi,esi,772                esi = Name * 0x772
0045B8AD    8BD6            mov edx,esi                     edx = Name * 0x772
0045B8AF    0FAFD6          imul edx,esi                    edx = (Name * 0x772) * (Name * 0x772)
0045B8B2    03F2            add esi,edx                     esi = ((Name * 0x772) * (Name * 0x772)) + (Name * 0x772)
0045B8B4    0BF6            or esi,esi                      같은 값을 or하면 똑같은 값이 나오니 이건 무시해도 됨
0045B8B6    69F6 74040000   imul esi,esi,474                esi = (((Name * 0x772) * (Name * 0x772)) + (Name * 0x772)) * 0x474
0045B8BC    03F6            add esi,esi                     esi = esi + esi
0045B8BE    8BD6            mov edx,esi                     edx = esi
0045B8C0    41              inc ecx
0045B8C1    48              dec eax
```

정리한 연산 코드이다. 특정 Name을 입력 했을때, EDX에는 문제의 Key였던 BEDA???? 이런 수가 들어 있을 것이다.
이것을 바탕으로 코드를 짜보면,

```
#include <stdio.h>

int main(void) {
    for (int i = 33; i < 134; ++i) {
        if (((((((i * 0x772) * (i * 0x772)) + (i * 0x772)) * 0x474) + ((((i * 0x772) * (i * 0x772)) + (i * 0x772)) * 0x474)) & 0xffff0000) >> 16) == 0xBEDA){
            printf( format: "%c", i);
        }
    }
}
```

```
main
test
C:\Users\a02\Desktop\project\test\cmake-build-debug\test.exe
F
```

대문자 "F" 가 나오는 것을 알 수 있다.

문제의 정답은 Name의 MD5였기 때문에 "F"의 MD5를 구해서 입력해주면,

여기 MD5 해시하고자하는 텍스트를 붙여 넣습니다

F

MD5 해시를 생성!

당신의 MD5 메시지 여기에서 소화 복사합니다.

8006189430253l5F869E4E1F09471012

통과하게 된다.

이번 문제는 좀 헷갈렸다.