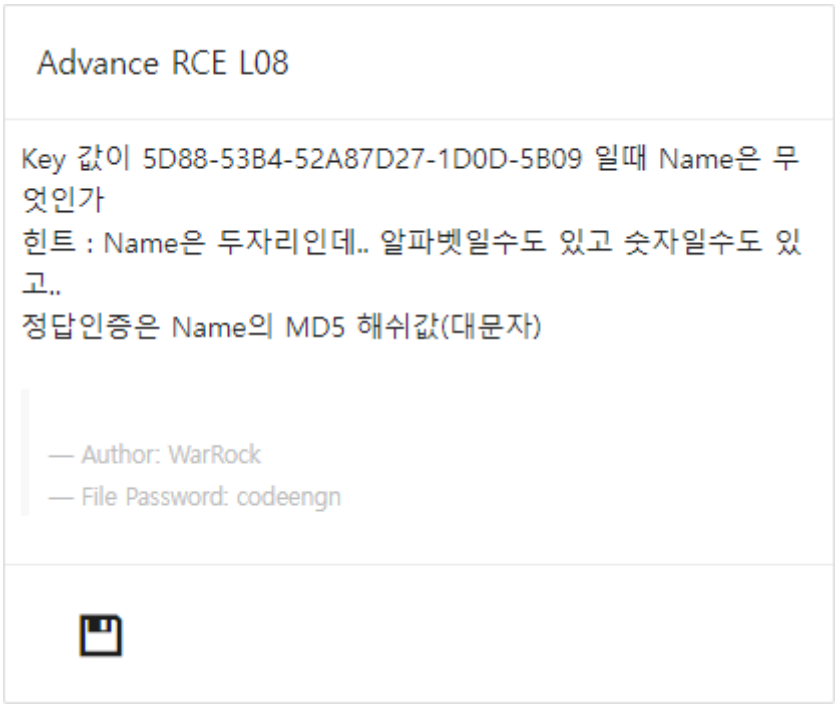
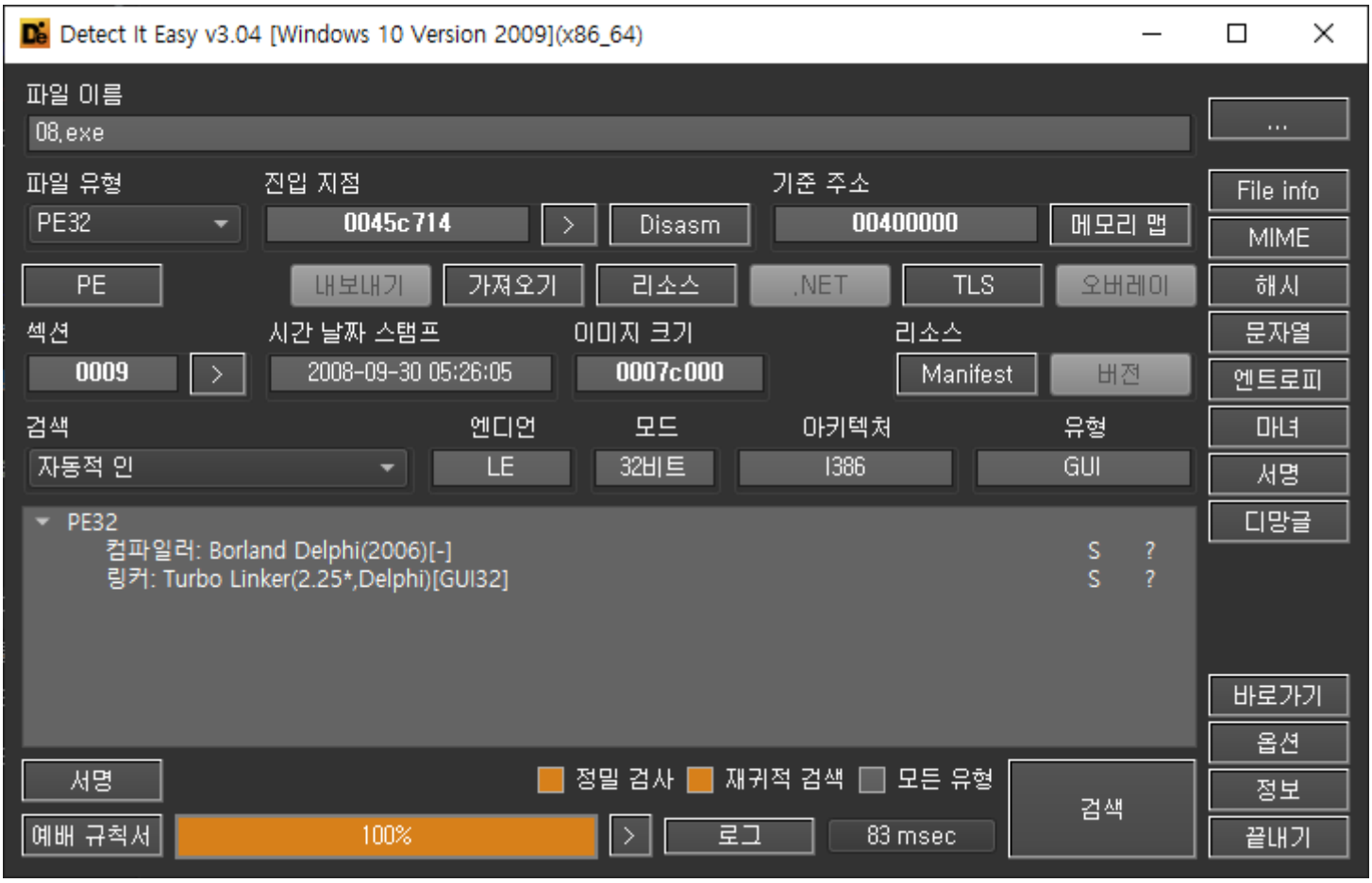
codeengn-advance-L08 풀이

리버싱 문제풀이 / Wonlf / 2022. 5. 12. 22:00



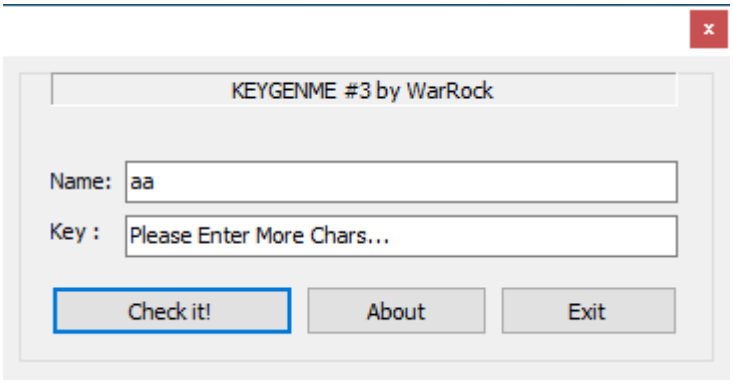
문제는 특정 시리얼 일 때, Name을 원하고 있다.

Die로 열어본다.



특이 사항은 보이지 않는다.

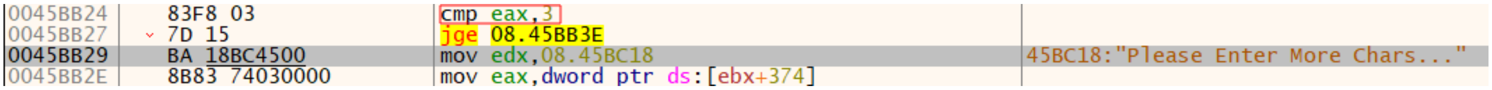
프로그램을 실행시켜본다.



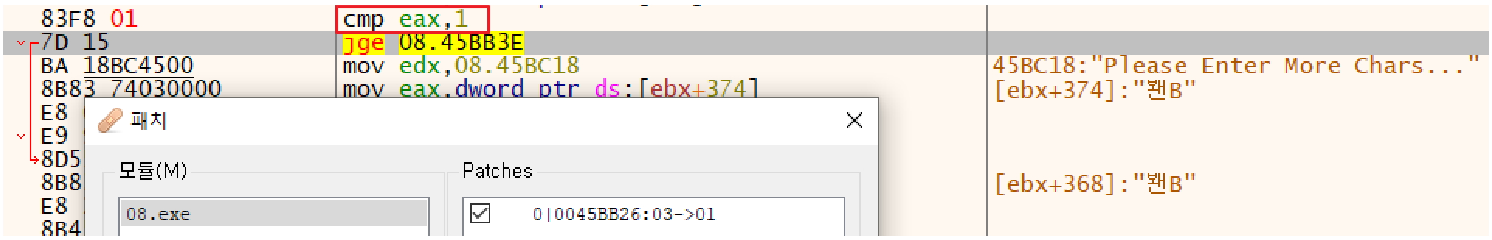
친절하게 Name과 Key를 알려주고 있고, 문제에서는 여기를 말하는 것 같다.

하지만 문제에서는 Name이 2글자라고 했으나, 2글자를 입력하면 Key에 특정 문자열이 들어가는 것을 보니 우회를 해주어야 할 거 같다.

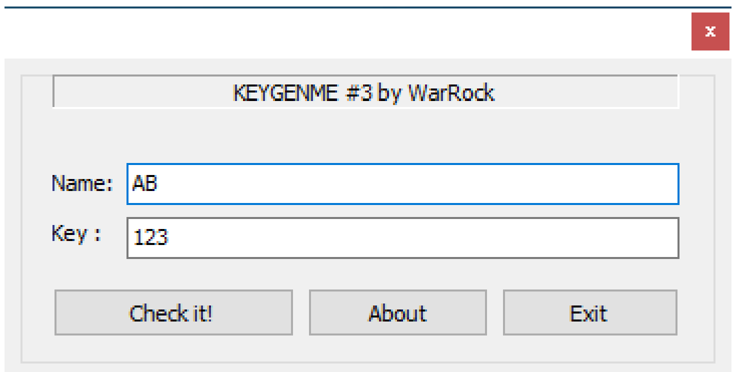
디버거로 열어본다.



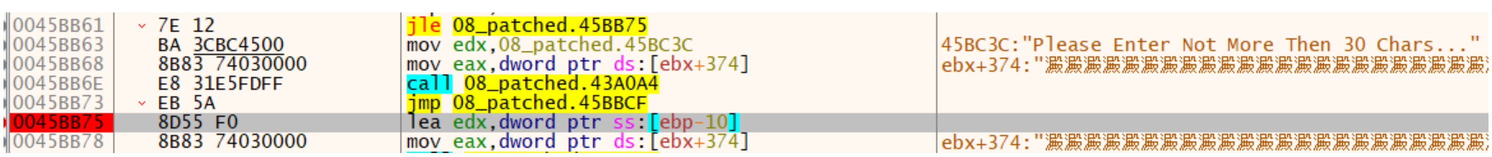
문자열 찾기로 비교하는 구문을 찾았다. 문자열의 길이가 3자리 보다 크지 않으면 특정 문자열을 출력했다.



cmp eax, 3 -> cmp eax, 1로 바꾸어주고 패치하여 새로운 exe로 저장한다.



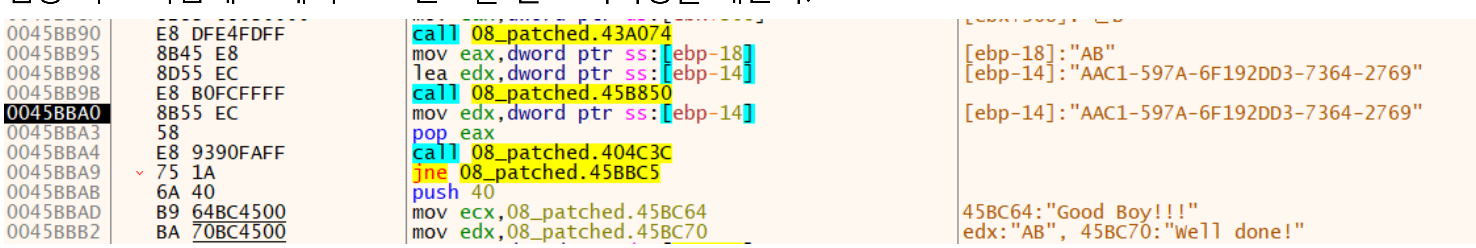
2자리 문자열 AB와 임의의 키 12345를 입력하고 디버깅을 시작해본다.



확인해보니, Name의 길이가 30이상 이라면 또 텍스트가 출력되는 구문이 있었다.

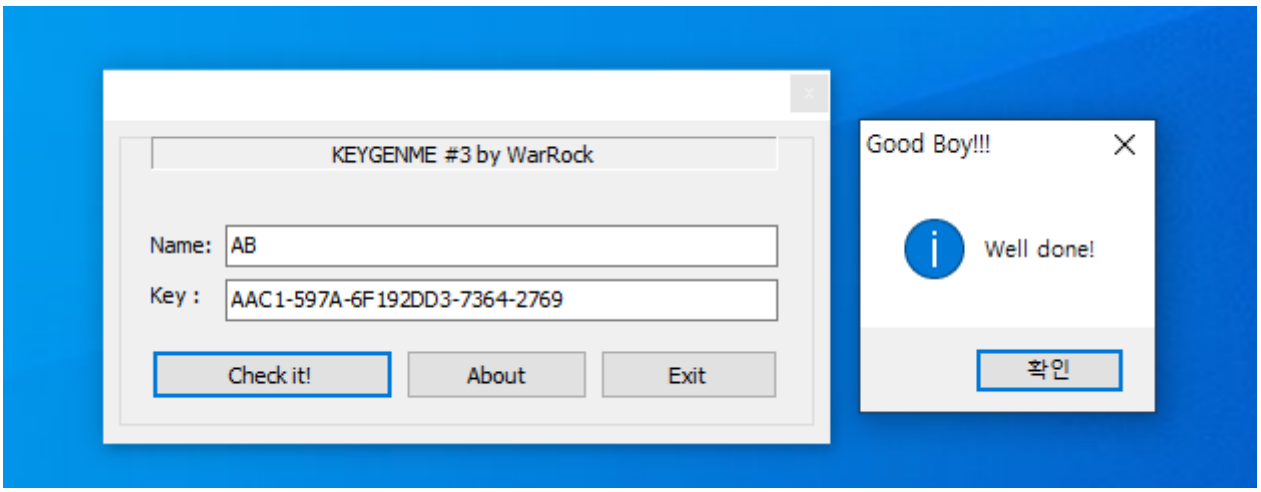
총 2단계의 검증을 거치고 난 뒤에 오는 구문이 비교하는 구문이 시작되는 부분일 것이다.

검증 바로 다음에 브레이크 포인트를 걸고 디버깅을 해본다.



아래로 내려가보니 성공을 출력하는 구문이 있었고, Key로 보이는 특정 문자열이 보였다.

이것을 실제 프로그램에 대입시켜보면,



실제로 맞다는 것을 알 수 있었다.

```
0045BB98   8D55 EC        lea edx,dword ptr ss:[ebp-14]   [ebp-14]:"AAC1-597A-6F192DD3-7364-2769"
0045BB9B   E8 B0FCFFFF    call 08_patched.45B850
0045BBA0   8B55 EC        mov edx,dword ptr ss:[ebp-14]   [ebp-14]:"AAC1-597A-6F192DD3-7364-2769"
```

확인 해보니, 45B850 함수를 호출하고 나면, Name에 따라 Key가 생성되는 것을 확인 할 수 있었다.

이 함수를 뜯어본다면, Key를 가지고 Name을 얻는 역연산 코드를 알 수 있을 것이다.
확인해본다.

함수에 들어오면 첫번째로 만나는 연산 구문이다.

```
0045B896   7E 2C          jle 08_patched.45B8C4
0045B898   B9 01000000    mov ecx,1                       ecx = 1
0045B89D   8B5D FC        mov ebx,dword ptr ss:[ebp-4]    ebx = "AB"
0045B8A0   0FB6740B FF    movzx esi,byte ptr ds:[ebx+ecx-1]  esi = ebx[ecx - 1]
0045B8A5   03F2           add esi,edx                     esi + ? 전에 연산했던  값과 더함
0045B8A7   69F6 72070000  imul esi,esi,772                esi = (esi + 0) * 0x772
0045B8AD   8BD6           mov edx,esi                     edx = esi
0045B8AF   0FAFD6         imul edx,esi                    edx = edx * esi
0045B8B2   03F2           add esi,edx                     esi = esi + edx
0045B8B4   0BF6           or esi,esi                      esi = esi | esi
0045B8B6   69F6 74040000  imul esi,esi,474                esi = esi * 0x474
0045B8BC   03F6           add esi,esi                     esi = esi + esi
0045B8BE   8BD6           mov edx,esi                     edx = esi
0045B8C0   41             inc ecx
0045B8C1   48             dec eax
```

2자리의 Name을 한자리씩 연산하는 구문이다.

이것을 C코드로 구현해놓고 확인해보면,

```
FFE374F0
AAC16C20
```

Key의 첫번째 값(AAC1)이 포함되어 있는 것을 알 수 있다.

이것만으로도 Name을 구할 수 있겠지만 아래 구문을 봐버려서 아래 구문도 해석해보겠다.

두번째로 만나게 되는 연산 구문이다.

```
0045B8D5    8B55 FC          mov edx,dword ptr ss:[ebp-4]          [ebp-4]:"AB"
0045B8D8    0FB65402 FF      movzx edx,byte ptr ds:[edx+eax-1]     이번엔 역순으로
0045B8DD    83C2 11          add edx,11                            edx = edx[i] + 0x11
0045B8E0    83EA 05          sub edx,5                             edx = edx - 5
0045B8E3    69D2 92000000    imul edx,edx,92                       edx = edx * 92
0045B8E9    03D2             add edx,edx                           edx = edx + edx
0045B8EB    69D2 19080000    imul edx,edx,819                      edx = edx * 0x819
0045B8F1    0155 F0          add dword ptr ss:[ebp-10],edx         첫번째 연산값과 두번째 연산값 더하기
```

이것 또한 C코드로 구현하고 확인해보면,

```
2D07038
597A3EC
```

Key의 두번째 값(597A)이 포함되어 있는 것을 알 수 있다.

이 2가지 조건을 바탕으로 역연산 코드를 작성해보겠다.

```
#include <stdio.h>

unsigned int first(int val[]){
    int esi, edx = 0;

    for (int i = 0; i < 2; i++) {

        esi = val[i];
        esi = esi + edx;
        esi = esi * 0x772;
        edx = esi;
        edx = edx * esi;
        esi = esi + edx;
        esi = esi | esi;
        esi = esi * 0x474;
        esi = esi + esi;
        edx = esi;


    }
    return (edx & 0xffff0000) >> 16;
}

unsigned int second(int val[]){
    int edx = 0;
    int ebp = 0;

    for (int i = 1; i >= 0; i--) {
        edx = val[i] + 0x11;
        edx = edx - 5;
        edx = edx * 0x92;
```

```
        edx = edx + edx;
        edx = edx * 0x819;
        ebp = ebp + edx;
    }
    return (ebp & 0xfffff000) >> 12;
}

int main(void) {

    for (int i = 33; i < 134; ++i) {
        for (int j = 33; j < 134; ++j) {
            int value[2] = {i, j};
            unsigned int a = first(value);
            unsigned int b = second(value);

            if(a == 0x5D88 && b == 0x53B4){
                printf("%c%c", i, j);
            }
        }
    }

    return 0;
}
```

원하는 Name을 얻었고 이것을 MD5로 바꾸어주고 인증해주면...

성공!