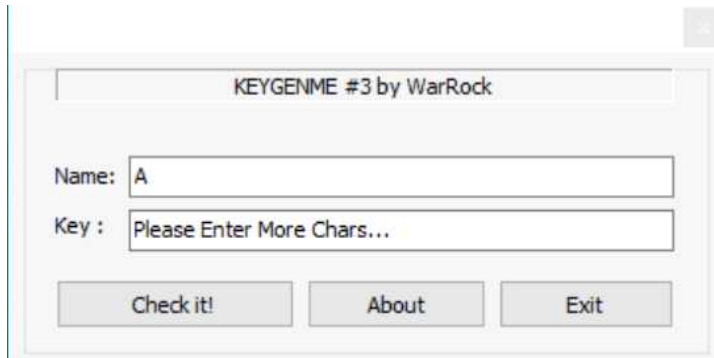


CodeEngn Basic RCE L17

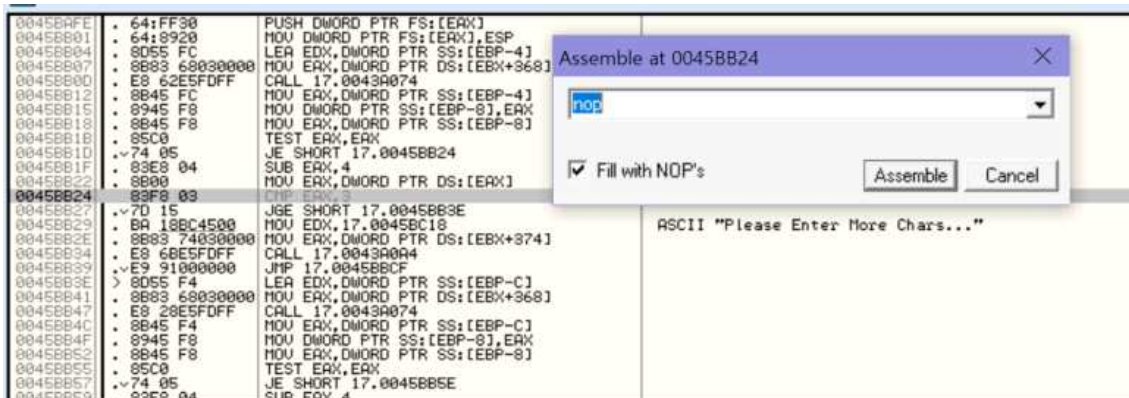
: Key 값이 BEDA-2F56-BC4F4368-8A71-870B 일때 Name은 무엇인가

힌트 : Name은 한자리인데.. 알파벳일수도 있고 숫자일수도 있고..

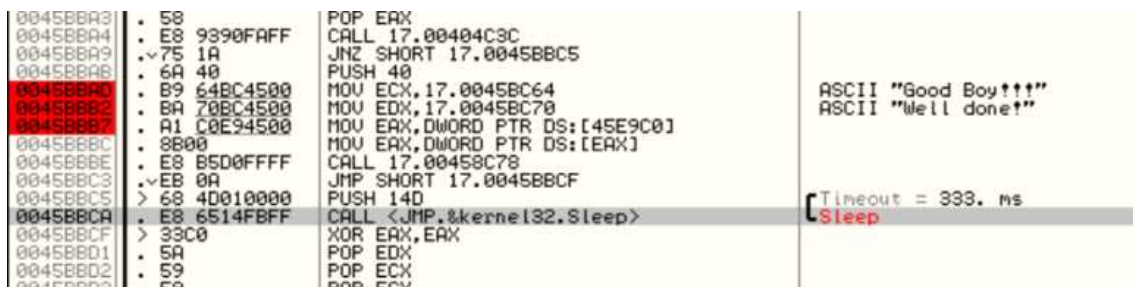
정답인증은 Name의 MD5 해쉬값(대문자)



name은 한자리라고 해서 아무거나 입력하고 눌렀는데 더 입력하라고 나온다  
이 부분은 올리디버거를 열어 수정해주도록 해야겠다 !



text string 찾기로 해당 메시지를 찾아서 이동했고, cmp eax,3으로 보아 3글자가 아니면 더  
입력하라고 하는 것 같다. 이 부분을 nop으로 바꾸어준다.



조금만 밑으로 내려가다 보면 성공메세지가 출력되는 부분을 볼 수 있다. 코드를 살펴보면 위  
에 JNZ 0045BBC5가 있는데 이 주소는 Timeout으로 프로그램이 종료되도록 하는 것 같다.

즉 JNZ를 그냥 통과해야 성공메세지가 출력이 되는 분기점이기 때문에 CALL 00404C3C 부분으로 이동해서 살펴보아야 겠다.

|          |         |                                |
|----------|---------|--------------------------------|
| 00404C39 | 8040 00 | LEA EAX,DWORD PTR DS:[EAX]     |
| 00404C3C | 39D0    | CMP EAX,EDX                    |
| 00404C3E | 74 30   | JE SHORT 17.00404C70           |
| 00404C40 | 85D0    | TEST EAX,EDX                   |
| 00404C42 | 74 40   | JE SHORT 17.00404C84           |
| 00404C44 | 0FB608  | MOVZX ECX,BYTE PTR DS:[EAX]    |
| 00404C47 | 2A0A    | SUB CL,BYTE PTR DS:[EDX]       |
| 00404C49 | 75 25   | JNZ SHORT 17.00404C70          |
| 00404C4B | 53      | PUSH EBX                       |
| 00404C4C | 8B58 FC | MOV EBX,DWORD PTR DS:[EAX-4]   |
| 00404C4F | 2B5A FC | SUB EBX,DWORD PTR DS:[EDX-4]   |
| 00404C52 | 53      | PUSH EBX                       |
| 00404C53 | 83D1 FF | ADC ECX,-1                     |
| 00404C56 | 21D9    | AND ECX,EBX                    |
| 00404C58 | 2B48 FC | SUB ECX,DWORD PTR DS:[EAX-4]   |
| 00404C5B | 29C8    | SUB EAX,ECX                    |
| 00404C5D | 29CA    | SUB EDX,ECX                    |
| 00404C5F | 8B1C01  | MOV EBX,DWORD PTR DS:[ECX+EAX] |
| 00404C62 | 331C11  | XOR EBX,DWORD PTR DS:[ECX+EDX] |
| 00404C65 | 75 0A   | JNZ SHORT 17.00404C71          |
| 00404C67 | 83C1 04 | ADD ECX,4                      |
| 00404C6A | 78 F3   | JS SHORT 17.00404C5F           |
| 00404C6C | 58      | POP EAX                        |
| 00404C6D | 01C0    | ADD EAX,EAX                    |

CMP EAX, EDX

JE 00404C70

= EAX - EDX를 한 값이 0이면, 00404C70으로 JUMP하라

그럼 EAX와 EDX의 값이 같다고 가정하고 00404C70으로 가보겠다.

|          |         |                              |
|----------|---------|------------------------------|
| 00404C6D | 01C0    | ADD EAX,EAX                  |
| 00404C6F | 5B      | POP EBX                      |
| 00404C70 | C3      | RETN                         |
| 00404C71 | 0FBCDB  | BSF EBX,EBX                  |
| 00404C74 | C1EB 03 | SHR EBX,3                    |
| 00404C77 | 01D9    | ADD ECX,EBX                  |
| 00404C79 | 79 F1   | JNS SHORT 17.00404C6C        |
| 00404C7B | 8A0401  | MOV AL,BYTE PTR DS:[ECX+EAX] |
| 00404C7E | 3A0411  | CMP AL,BYTE PTR DS:[ECX+EDX] |
| 00404C81 | 5B      | POP EBX                      |
| 00404C82 | 5B      | POP EBX                      |
| 00404C83 | C3      | RETN                         |
| 00404C84 | 85C0    | TEST EAX,EAX                 |
| 00404C86 | 74 08   | JE SHORT 17.00404C90         |
| 00404C88 | 85D2    | TEST EDX,EDX                 |
| 00404C8A | 75 B8   | JNZ SHORT 17.00404C44        |
| 00404C8C | 3950 FC | CMP DWORD PTR DS:[EAX-4],EDX |
| 00404C8F | C3      | RETN                         |

바로 RETN으로 0을 리턴한다. 즉 CALL 00404C3C는 EAX와 EDX를 비교해 같으면 0을 리턴하는 것으로 볼 수 있다.

| Registers (FPU) |          |                                 |
|-----------------|----------|---------------------------------|
| EAX             | 023F8E08 | ASCII "BEDA-2F56-BC4F4368-8A71" |
| ECX             | 000000FC |                                 |
| EDX             | 023F8E38 | ASCII "FFE3-2C73-0502A34C-8A48" |
| EBX             | 02397170 |                                 |
| ESP             | 0019F2F4 |                                 |
| EBP             | 0019F31C |                                 |
| ESI             | 0042A3F0 | 17.0042A3F0                     |
| EDI             | 0019F48C |                                 |
| EIP             | 0045BBA9 | 17.0045BBA9                     |
| C               | 1        | ES 002B 32bit 0(FFFFFFFF)       |
| P               | 1        | CS 0023 32bit 0(FFFFFFFF)       |

JNZ에 BP를 걸어주고 실행시켜 NAME에는 'A', KEY에 BEDA-2F56-BC4F4368-8A71-870B를 눌러 CHECK IT을 누르면 EAX에 내가 입력한 시리얼 넘버가 들어간다. 아마도 이름이 A일때의 KEY는 FFE3-2C73-052A34C-8A48인 것을 추측할 수 있다.

|          |               |                               |                     |
|----------|---------------|-------------------------------|---------------------|
| 0045B890 | . E8 DFE4DFFF | CALL 17.0043A074              |                     |
| 0045B895 | . 8B45 E8     | MOV EAX,DWORD PTR SS:[EBP-18] |                     |
| 0045B898 | . 8D55 EC     | LEA EDX,DWORD PTR SS:[EBP-14] |                     |
| 0045B89B | . E8 B0FCFFFF | CALL 17.0045B850              |                     |
| 0045B8A0 | . 8B55 EC     | MOV EDX,DWORD PTR SS:[EBP-14] |                     |
| 0045B8A3 | . 58          | POP EAX                       |                     |
| 0045B8A4 | . E8 9390FAFF | CALL 17.00404C3C              |                     |
| 0045B8A7 | . 75 1A       | JNZ SHORT 17.0045B8C5         |                     |
| 0045B8AB | . 6A 40       | PUSH 40                       |                     |
| 0045B8AD | . B9 64BC4500 | MOV ECX,17.0045BC64           | ASCII "Good Boy!!!" |
| 0045B8B2 | . BA 70BC4500 | MOV EDX,17.0045BC70           | ASCII "Well done!"  |
| 0045B8B7 | . A1 C0E94500 | MOV EAX,DWORD PTR DS:[45E9C0] |                     |
| 0045B8BC | . 8B00        | MOV EAX,DWORD PTR DS:[EAX]    |                     |
| 0045B8BE | . E8 B5D0FFFF | CALL 17.0045BC78              |                     |
| 0045B8C3 | . EB 0A       | JMP SHORT 17.0045B8CF         |                     |
| 0045B8C5 | . 68 4D010000 | PUSH 140                      |                     |
| 0045B8CA | . E8 6514FBFF | CALL <JMP.&kernel32.Sleep>    | Timeout = 333. ms   |
| 0045B8CF | . 33C0        | XOR EAX,EAX                   | Sleep               |
| 0045B8D1 | . 5A          | POP EDX                       |                     |

그럼 CALL 00404C3C는 사용자의 입력값과 시리얼값을 비교하는 함수를 호출하고, 바로 그 위에 호출되는 함수가 NAME에 대한 시리얼 값을 생성하는 함수라고 추측할 수 있으므로 0045B850으로 이동해서 살펴보겠다.

|          |                 |                                   |  |
|----------|-----------------|-----------------------------------|--|
| 0045B894 | > 85C0          | TEST EAX,EAX                      |  |
| 0045B896 | . 7E 2C         | JLE SHORT 17.0045B8C4             |  |
| 0045B898 | . B9 01000000   | MOV ECX,1                         |  |
| 0045B89D | > 8B5D FC       | MOV EBX,DWORD PTR SS:[EBP-4]      |  |
| 0045B8A0 | . 0FB6740B FF   | MOVZX ESI,BYTE PTR DS:[EBX+ECX-1] |  |
| 0045B8A5 | . 03F2          | ADD ESI,EDX                       |  |
| 0045B8A7 | . 69F6 72070000 | IMUL ESI,ESI,772                  |  |
| 0045B8AD | . 8BD6          | MOV EDX,ESI                       |  |
| 0045B8AF | . 0FAFD6        | IMUL EDX,ESI                      |  |
| 0045B8B2 | . 03F2          | ADD ESI,EDX                       |  |
| 0045B8B4 | . 0BF6          | OR ESI,ESI                        |  |
| 0045B8B6 | . 69F6 74040000 | IMUL ESI,ESI,474                  |  |
| 0045B8BC | . 03F6          | ADD ESI,ESI                       |  |
| 0045B8BE | . 8BD6          | MOV EDX,ESI                       |  |
| 0045B8C0 | . 41            | INC ECX                           |  |
| 0045B8C1 | . 48            | DEC EAX                           |  |
| 0045B8C2 | . 75 D9         | JNZ SHORT 17.0045B89D             |  |
| 0045B8C4 | > 8B45 FC       | MOV EAX,DWORD PTR SS:[EBP-4]      |  |
| 0045B8C7 | . 85C0          | TEST EAX,EAX                      |  |

0045B89D ~ 0045B8C2를 살펴보면

```

MOV EBX, DWORD PTR SS:[EBP-4]
MOVZX ESI, BYTE PTR DS:[EBX+ECX-1]
ADD ESI, EDX // ESI = ESI + EDX
IMUL ESI, ESI, 772 // ESI = ESI * 772
MOV EDX, ESI // EDX = ESI
IMUL EDX, ESI // EDX = EDX * ESI
ADD ESI, EDX // ESI = ESI + EDX
OR ESI, ESI // ESI = ESI
IMUL ESI, ESI, 474 // ESI = ESI * 474
ADD ESI, ESI // ESI = ESI + ESI
MOV EDX, ESI // EDX = ESI
INC ECX // ECI + 1
DEC EAX // EAX - 1
JNZ 0045B89D
JNZ로 다시 맨 위로 올라가 루프를 돌며 생성하는 것 같았다 !
c++로 작성해 보고, 돌려보았다.

```

```

#include <stdio.h>

```

```

int main()
{
    int ESI, EDX = 0;

    for (int i = 0x30; i <= 0x7A; i++) { // 0에서 z 까지 ( ASCII 코드로 0x30 = 0,
0x7A = z )
        ESI += EDX; // ADD ESI, EDX
        ESI = i * 0x772; //IMUL ESI, ESI, 0x772
        EDX = ESI; //MOV EDX, ESI
        EDX *= ESI; //IMUL EDX, ESI
        ESI += EDX; //ADD ESI, EDX
        ESI = ESI; // OR ESI, ESI
        ESI *= 0x474; //IMUL ESI, ESI, 0x474
        ESI += ESI; //ADD ESI, ESI
        EDX = ESI; // MOV EDX, ESI

        printf("NAME = %c , PW = %X\n", i, EDX);
    }
    return 0;
}

```

+

IMUL : 인자가 2개일 경우, 첫번째 인자와 두번째 인자를 곱한 결과를 첫번째 인자에 저장한다. 인자가 3개일 경우, 두번째 인자와 세번째 인자를 곱한 결과를 첫번째 인자에 저장한다.

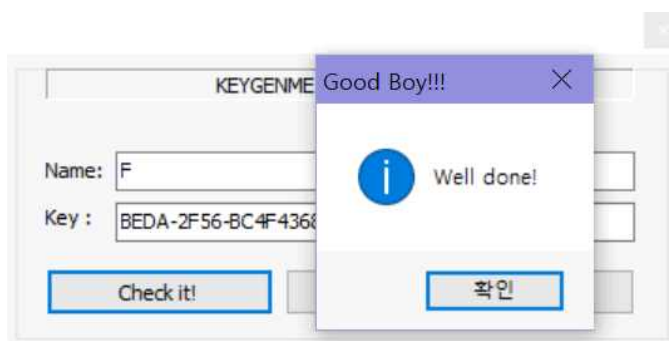
MOV DEST, SRC : SRC에 있는 값을 DEST에 저장

INC : 피연산자 + 1

DEC : 피연산자 - 1

```
NAME = 4 , PW = BAF14640
NAME = 5 , PW = 396AC030
NAME = 6 , PW = 93493D60
NAME = 7 , PW = C88CBDD0
NAME = 8 , PW = D9354180
NAME = 9 , PW = C542C870
NAME = : , PW = 8CB552A0
NAME = : , PW = 2F8CE010
NAME = < , PW = ADC970C0
NAME = = , PW = 76B04B0
NAME = > , PW = 3C719BE0
NAME = ? , PW = 4CDD3650
NAME = @ , PW = 38ADD400
NAME = A , PW = FFE374F0
NAME = B , PW = A27E1920
NAME = C , PW = 207DC090
NAME = D , PW = 79E26B40
NAME = E , PW = AEAC1930
NAME = F , PW = BEDACA60
NAME = G , PW = AA6E7ED0
NAME = H , PW = 71673680
```

그러면 F일 때 PW가 BEDA로 시작하는 것을 보아 F가 NAME인 것을 알 수 있다.



성공메세지가 출력된다