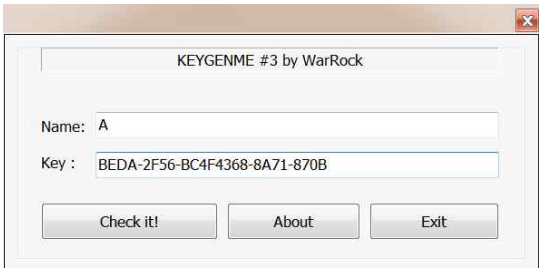
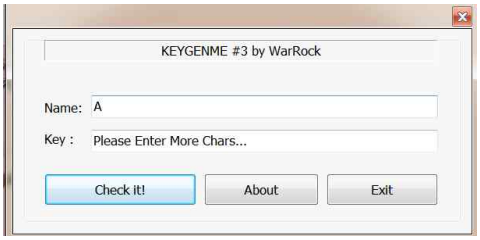
## 코드 엔진 Challenges: Basic 17

Author: WarRock
Korean: Key 값이 BEDA-2F56-BC4F4368-8A71-870B 일때 Name은 무엇인가
힌트 : Name은 한자리인데.. 알파벳일수도 있고 숫자일수도 있고..
정답인증은 Name의 MD5 해쉬값(대문자)

문제는 키 값에 따라 달라지는 네임 값을 구하는 것을 목표로 한다.




주어진 키 값을 입력하고 임의의 네임 A를 입력한 후 Check it을 누르면 다음과 같이 "Please~"란 문자열을 볼 수 있다. 더 많은 문자를 입력하라는 메시지 정보만 출력되고 DE로 분석하였을 때 패킹도 되어 있지 않다. 올리디버거로 분석해보자.

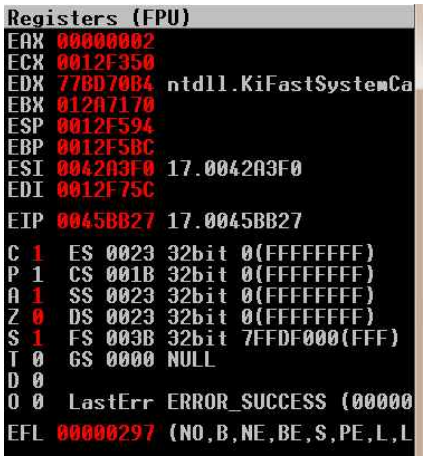
```
Address   Disassembly                Text string
0045B7ED  ASCII "Button2Click"
0045B800  ASCII "Button3Click"
0045B80D  ASCII "TForm1"
0045B836  ASCII "TForm1"
0045B847  ASCII "kg070"
0045BAE4  ASCII "-",0
0045BB29  MOV EDX,17.0045BC18      ASCII "Please Enter More Chars..."
0045BB63  MOV EDX,17.0045BC3C      ASCII "Please Enter Not More Then 30 Chars..."
0045BBAD  MOV ECX,17.0045BC64      ASCII "Good Boy!!!"
0045BBB2  MOV EDX,17.0045BC70      ASCII "Well done!"
0045BC18  ASCII "Please Enter Mor"
```

해당 문제는 key값이 "BEDA-2F56-BC4F4368-8A71-870B" 1자리 값의 Name을 구하는 것인데 한자리만 넣으면 "please~"라는 문자열이 출력되기에 이를 확인하기 위해 사용되는 문자열 목록에서 아까 본 메시지를 찾아 해당 주소로 이동해보자.

```
0045BB1F  . 83E8 04          SUB EAX,4
0045BB22  . 8B00             MOV EAX,DWORD PTR DS:[EAX]
0045BB24  > 83F8 03          CMP EAX,3
0045BB27  .v7D 15            JGE SHORT 17.0045BB3E
0045BB29  . BA 18BC4500      MOV EDX,17.0045BC18        ASCII "Please Enter More Chars..."
0045BB2E  . 8B83 74030000    MOV EAX,DWORD PTR DS:[EBX+374]
0045BB34  . E8 6BE5FDFF      CALL 17.0043A0A4
0045BB39  .vE9 91000000      JMP 17.0045BBCF
```

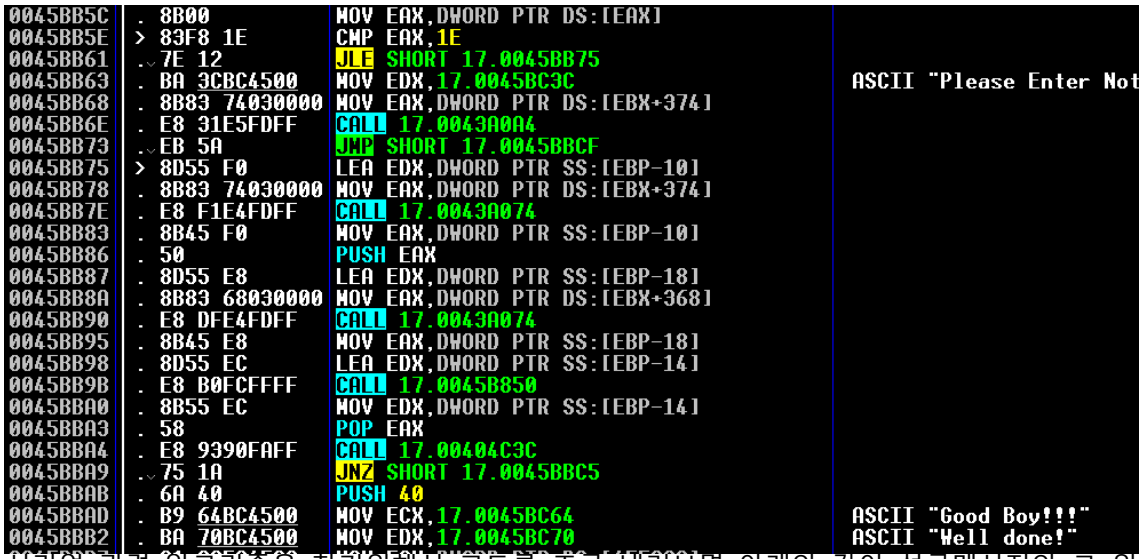
0045BB29로 이동했더니 두 줄 위의 코드에서 비교구문과 분기문을 확인 할 수있다. Name란에 입력된 글자 수를 제어하는 구문으로 추축된다. 해당 구문에 BP를 걸고 임의의 값을 입력한 뒤 EAX 값을 확인해보자.

**JGE는 Jump If greater or equal의 줄임말로 '>='를 의미한다. 크거나 같을 때 점프한다.
이름에 A도 넣어보고 AB도 넣어본 결과 EAX 값은 Name에 입력한 길이의 값을 저장한다.




Name에 "AB"를 입력했을 때의 EAX 레지스터의 값

해당 문제는 Name에 입력한 값의 길이가 1글자일때이므로 CMP EAX,1로 패치해준다.코드를 수정해준 뒤 ,바이너리 패치 본을 만들어 준다. 이제 무슨 Name이 맞는지 맞혀보자.
무차별 공격을 수행해도 되지만 시리얼 관련 알고리즘을 찾아보자.




시리얼 관련 알고리즘을 찾기위해서 코드를 조금 내려보면 아래와 같이 성공메시지와 그 위에 시리얼 관련 코드가 있는 것을 알 수있다. 그중에서도 0045B850 함수가 시리얼 생성 관련 함수 인 것을 알 수있다. 한 줄씩 코드를 실행시켜 보면 확인할 수있아 0045B850 함수로 들어가보면 아래와 같은 루틴이 반복되며 키를 생성 하는 값을 알 수있다 .그리고 그렇게 생성된 값들이 ESI 값에 저장된다는 것도 알 수 있다.

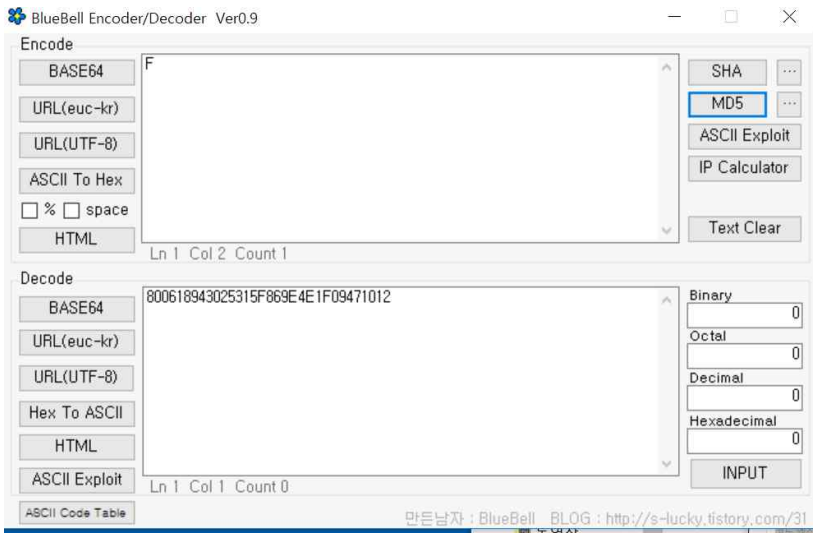
일단 위의 그림을 반복했을 때 ESI 값을 확인하면 이 값을 마친 후 생성된 시리얼 EDX값과 비쇼해봤을 ESI 값의 4자리값을 사용된다는 것을 알 수있다.
4자리값이 사용된다는 것을 알았으니 이제 위의 연산을 한번 진행해서 BEDAXXXX 값이 나오는 문자를 찾아 NAME 값을 찾아보자 .

※다음은 반복연산을 분석해 본 것이다.
MOV EBX,DWORD PTR SS:[EBP-4]
MOVZX ESI,BYTE PTR DS:[EBX+ECX-1] ESI 값에 입력값 대입 ESI = 61 (a)

```
ADD ESI,EDX      ESI = ESI + EDX (0)              //ESI = 61
IMUL ESI,ESI,772 ESI = ESI * 772                  //ESI = 2D232
MOV EDX,ESI      EDX = ESI                        //EDX = 2D232
IMUL EDX,ESI     EDX = EDX * ESI                  //EDX = 7F55E11C4
ADD ESI,EDX      ESI = ESI + EDX                  //ESI = 7F560E3F6
OR ESI,ESI
IMUL ESI,ESI,474 ESI = ESI * 474                  // ESI = 2370B3772378
ADD ESI,ESI      ESI = ESI + ESI                  // ESI = 46E166EE46F0
MOV EDX,ESI
EDX = ESI                          // EDX = 46E166EE46F0
INC ECX
DEC EAX
```

이제 위의 코드 생성 연산을 이용해 코드를 작성하며 BEDA와 비교하는 프로그램을 만들어서 값을 찾아보면 된다.

답은 F가 나오는 것을 확인할 수 있다.




F의 MD5 해쉬값은 이러하다. 8006189430253 15F869E4E1F09471012