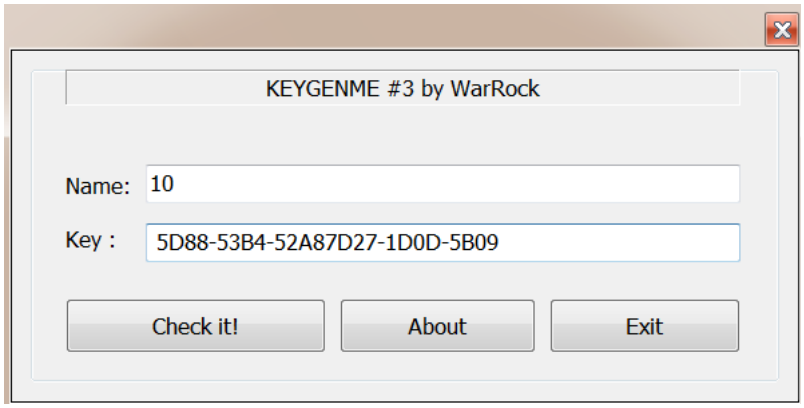
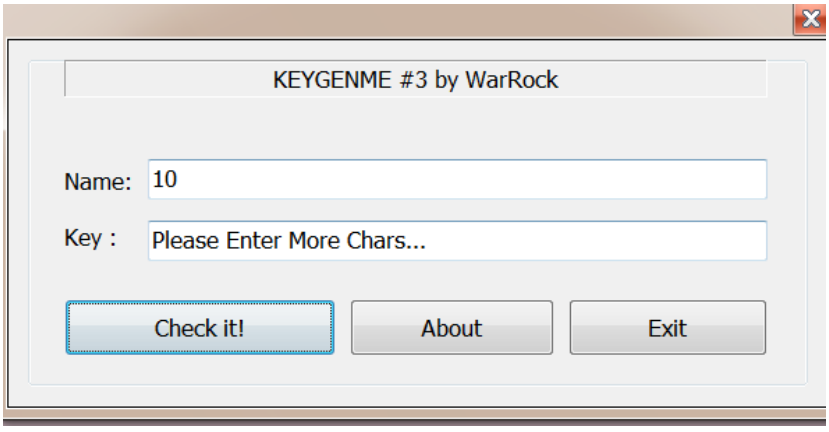
코드 엔진 Challenges: Advance 08

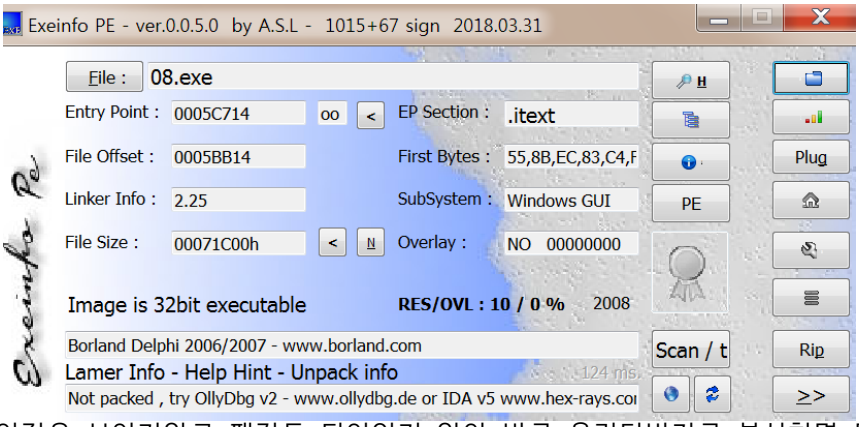
Author: WarRock
Korean: Key 값이 5D88-53B4-52A87D27-1D0D-5B09 일때 Name은 무엇인가
힌트 : Name은 두자리인데.. 알파벳일수도 있고 숫자일수도 있고..
정답인증은 Name의 MD5 해쉬값(대문자

문제는 주어진 킷값에 대한 Name을 찾는 것이다. 먼저 파일을 실행해 힌트에 나온대로 두자리 이지만 내가 알아보기 쉬운 '10'을 name에 넣고 키 값을 입력해보았다.





키 값이 "Pleas Enter More Chars.." 라는 문구로 바뀌었다. 파일을 살펴봤으니 PEID로 패킹여부로 특이점을 확인해보자.



별다른 특이점은 보이지않고 패킹도 되어있지 않아 바로 올리디버거로 분석하면 될 것 같다.

```
0045BAE4 ASCII "-",0
0045BB29 MOV EDX,08.0045BC18          ASCII "Please Enter More Chars..."
0045BB63 MOV EDX,08.0045BC3C          ASCII "Please Enter Not More Then 30 Chars..."
0045BBAD MOV ECX,08.0045BC64          ASCII "Good Boy!!!"
0045BBB2 MOV EDX,08.0045BC70          ASCII "Well done!"
0045BC18 ASCII "Please Enter Mor"
```

문자열 찾기를 이용하자 아까 본 문구와 함께 성공시 출력될 문자열도 보인다. 이 부분의 코드 영역으로 이동해 분기문이 있나 살펴보자.

```
0045BB0D . E8 62E5FDFF    CALL 08.0043A074
0045BB12 . 8B45 FC        MOV EAX,DWORD PTR SS:[EBP-4]
0045BB15 . 8945 F8        MOV DWORD PTR SS:[EBP-8],EAX
0045BB18 . 8B45 F8        MOV EAX,DWORD PTR SS:[EBP-8]
0045BB1B . 85C0           TEST EAX,EAX
0045BB1D .v74 05          JE SHORT 08.0045BB24
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 08.0045BB3E
0045BB29 . BA 18BC4500    MOV EDX,08.0045BC18          ASCII "Please Enter More Chars..."
0045BB2E . 8B83 74030000  MOV EAX,DWORD PTR DS:[EBX+374]
0045BB34 . E8 6BE5FDFF    CALL 08.0043A0A4
0045BB39 .vE9 91000000    JMP 08.0045BBCF
0045BB3E > 8D55 F4        LEA EDX,DWORD PTR SS:[EBP-C]
0045BB41 . 8B83 68030000  MOV EAX,DWORD PTR DS:[EBX+368]
0045BB47 . E8 28E5FDFF    CALL 08.0043A074
0045BB4C . 8B45 F4        MOV EAX,DWORD PTR SS:[EBP-C]
0045BB4F . 8945 F8        MOV DWORD PTR SS:[EBP-8],EAX
0045BB52 . 8B45 F8        MOV EAX,DWORD PTR SS:[EBP-8]
0045BB55 . 85C0           TEST EAX,EAX
0045BB57 .v74 05          JE SHORT 08.0045BB5E
0045BB59 . 83E8 04        SUB EAX,4
0045BB5C . 8B00           MOV EAX,DWORD PTR DS:[EAX]
0045BB5E > 83F8 1E        CMP EAX,1E
0045BB61 .v7E 12          JLE SHORT 08.0045BB75
0045BB63 . BA 3CBC4500    MOV EDX,08.0045BC3C          ASCII "Please Enter Not More Then 30 Chars..."
0045BB68 . 8B83 74030000  MOV EAX,DWORD PTR DS:[EBX+374]
```

코드를 대략적으로 본 결과 0045BB27의 JGE short 08.45BB3E가 밑의 "please Enter More Chars "를 건너뛰어 성공메시지로 가는 핵심 분기문인 것 같다. 이 곳에 bp를 걸어 파일을 실행하고 레지스터 값들을 살펴보자.

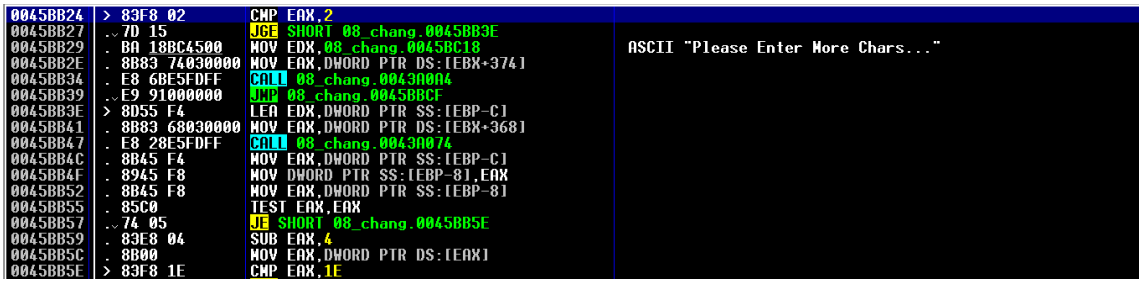
```
EAX 00000002
ECX 0012F350
EDX 777C70B4 ntdll.KiFastSystemCallRet
EBX 01377170
ESP 0012F594
EBP 0012F5BC
ESI 0042A3F0 08.0042A3F0
EDI 0012F75C

EIP 0045BB27 08.0045BB27
```

0045BB27에서 멈췄을때의 레지스터 값은 이러하다. 네임값에 '10'을 입력했으니 아마도 EAX의 값은 Name 의 길이를 저장하고 있는 것 같다. 더 정확하게 보기위해 그 전의 코드부터 살펴보도록 하자.

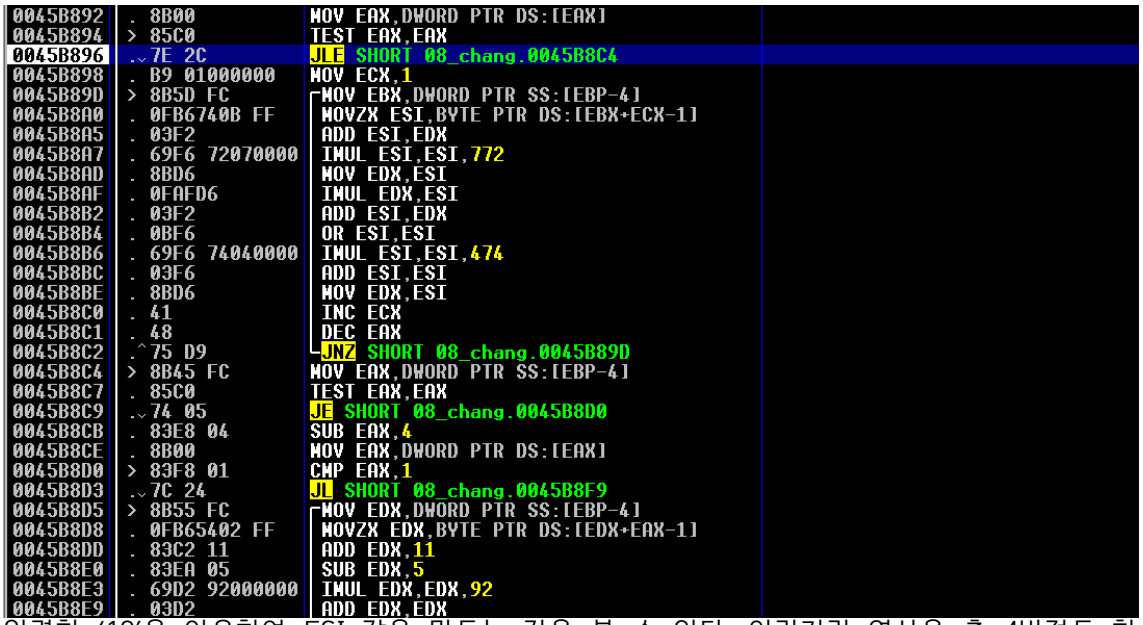
*JGE 결과가 크거나 같으면 분기

위의 0045BB24 CMP EAX,3에서 EAX의 크기가 3보다 크거나 같으면 0045BB3E로 분기하게 되는데 우리가 입력한 name 값은 2글자라 "please enter more chars"가 출력된다 그러면 우리가 구하고자하는 name값은 2글자다 그러니 이 부분의 코드를 수정하자 .

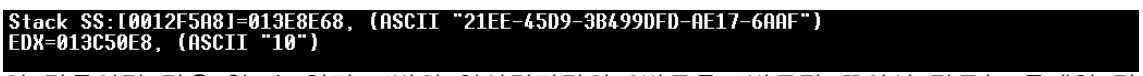


코드를 수정해 2글자여도 정상적으로 시리얼 생성부분에 갈 수 있게 하였다. 그리고 나서 계속 코드를 실행해보았따.

코드를 따라가다보면 0045BB9B 이후에 0045BBA0에서 우리가 입력한 시리얼 값이 아닌 다른 시리얼 값이 나오는 것을 알 수 있다. 0045BB9B가 시리얼값을 생성하는 것을 알 수 있는데 0045BB9B를 실행해 함수안으로 들어가보자.



입력한 '10'을 이용하여 ESI 값을 만드는 것을 볼 수 있다. 이런저런 연산을 총 4번정도 한 후 입력받은 name의 문자열을 갖고 key값을 만든다. 실행을 다 하면 10을 이용한 문자열



이 만든어진 것을 알 수 있다. 4번의 연산결과값의 8비트중 4비트만 뚜와서 만드는 문제인 것

을 알 수 있다.
정리를 해보면

```
mov esi ,입력한 값의 첫 번째 자릿수(1)
add esi,edx
imul esi,esi,0x772
mov edx,esi
imul edx,esi
add esi,edx
imul esi,esi,0x474
add esi,esi,
mov edx,esi
mov esi ,입력한 값의 두 번째 자릿수(0)
add esi,edx
imul esi,esi,0x772
mov edx,esi
imul edx,esi
add esi,edx
imul esi,esi,0x474
add esi,esi,
mov edx,esi
```

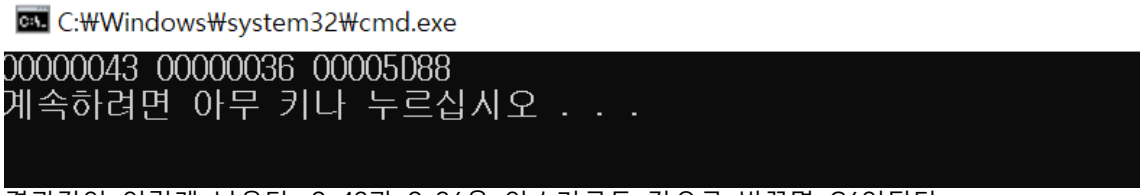
이렇게 구한 값의 ESI값의 상위 16비트값이 시리얼값의 첫부분 4자리값이다.

비쥬얼 스튜디오로 코딩을 해보면

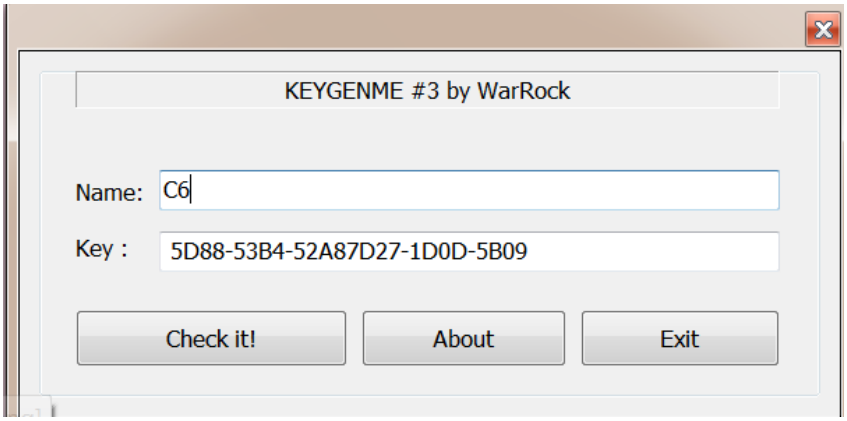
```
#include<stdio.h>
#include<Windows.h>
int main()
{

int arr[2], number; //arr에는 name값이 들어감 (2자리)
//arr[0] : 첫째 자리 ,arr[1] : 둘째 자리
int result;
for (int i = 0x30; i <=0x7A; i++)
 { //0x30:아스키코드 0 0x7A:아스키 코드 소문자 z
  //name에 들어갈 수 있는 모든 값들을 넣기위해
        arr[0] = i;
        for (int j  = 0x30; j <= 0x7A ; j++)
        {
                arr[1] = j;
                __asm {
```

```
                mov EDX, 0
            //add ESI,EDX 에서 첫자리수를 연살한때 EDX값이 0
        //두번째 자리 수를 연산 시에는 EDX값이 첫자리 수를 연산하고 남은 값이
        //저장되어 있어 일부러 초기화를 하지 않음
            }
            for (int k = 0; k < 2; k++)
            {
                number = arr[k];
                __asm
                {
                    mov ESI,number
                    add ESI,EDX
                    IMUL ESI,ESI,0x772
                    mov EDX,ESI
                    IMUL EDX ,ESI
                    ADD ESI,EDX
                    OR ESI, ESI
                    IMUL ESI, ESI, 0x474
                    ADD ESI,ESI
                    MOV EDX,ESI
                    }
                }
                __asm
                {
                    mov result,ESI
                }
                if (HIWORD(result) == 0x5D88)
                {
// name값을 넣고 위의 반복문을 돌려서 구한 ESI의 상위 16비트 값이 첫부분 4자리값이다.
//result의 상위 16비트 값이 문제에 적힌 시리얼의 첫 4자리값과 같은 경우
//이때의 name값들을 알기위해 16진수로 출력
                printf("%p %p %p \n", arr[0], arr[1], HIWORD(result));
                }
            }
        }
    return 0;
}
```



결과값이 이렇게 나온다. 0x43과 0x36을 아스키코드 값으로 바꾸면 C6이된다.
답은 C6이다



C6의 MD5
7E8B9F5CAB4A8FE24FAD9FE4B7452702