Codeengn Challenges Advance RCE LEVEL8 풀이

Reverse2 L08 Start

Author : WarRock

Korea :

Key 값이 5D88-53B4-52A87D27-1D0D-5B09 일때 Name은 무엇인가

힌트 : Name은 두자리인데.. 알파벳일수도 있고 숫자일수도 있고..

정답인증은 Name의 MD5 해쉬값(대문자)

English :

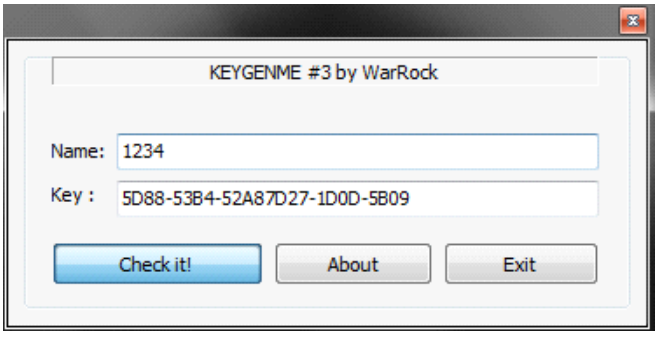
What is Name when the Key is 5D88-53B4-52A87D27-1D0D-5B09

Hint : The name is 2 letters and it could be either alphabetic or numeric.

Verify your solution with the MD5 value of the Name(in CAPITALS).

Down

먼저 프로그램을 실행시켜보도록 했습니다.



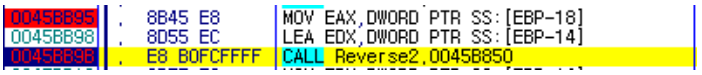
Name과 Key값을 입력 후 Check 하는 방식의 프로그램입니다.

PEID로 확인을 해보니 Delphi로 만들어진 프로그램이었습니다.
따로 패킹된것은 없으니 바로 올리로 분석을 해보기로 하였습니다.

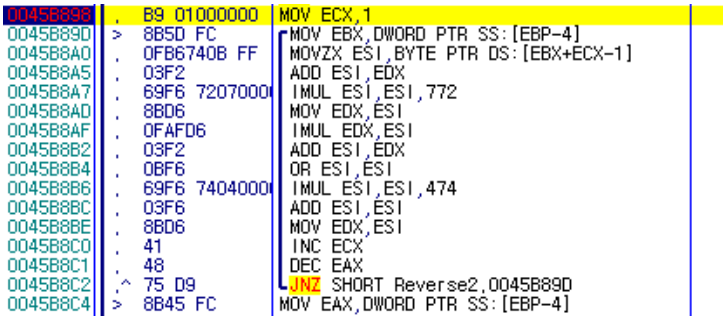
```
0045BB24 | > 83F8 03      | CMP EAX,3
0045BB27 | .v 7D 15       | JGE SHORT Reverse2.0045BB3E
0045BB29 | .  BA 18BC4500 | MOV EDX,Reverse2.0045BC18          ASCII "Please Enter More Chars..."
0045BB2E | .  8B83 740300 | MOV EAX,DWORD PTR DS:[EBX+374]
0045BB34 | .  E8 6BE5FDFF | CALL Reverse2.0043A0A4
0045BB39 | .v E9 91000000 | JMP Reverse2.0045BBCF
0045BB3E | >  8D55 F4     | LEA EDX,DWORD PTR SS:[EBP-C]
0045BB41 | .  8B83 680300 | MOV EAX,DWORD PTR DS:[EBX+368]
0045BB47 | .  E8 28E5FDFF | CALL Reverse2.0043A074
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 | .v 74 05       | JE SHORT Reverse2.0045BB5E
0045BB59 | .  83E8 04     | SUB EAX,4
0045BB5C | .  8B00        | MOV EAX,DWORD PTR DS:[EAX]
0045BB5E | >  83F8 1E     | CMP EAX,1E
0045BB61 | .v 7E 12       | JLE SHORT Reverse2.0045BB75
0045BB63 | .  BA 3CBC4500 | MOV EDX,Reverse2.0045BC3C          ASCII "Please Enter Not More Then 30 Chars..."
```

먼저 보이는 것은 3~30 사이의 글자인지 체크해주는 부분입니다.
저는 1234를 입력했으니 무사통과를 하였습니다 :D

그리고 분석을 하다가 1234를 Name으로 입력하면 Key는 A672-9051-FE90F705-8EA9-BBD4가 되는걸 알게되었습니다. 그리고 다시 분석을 진행해보았습니다.

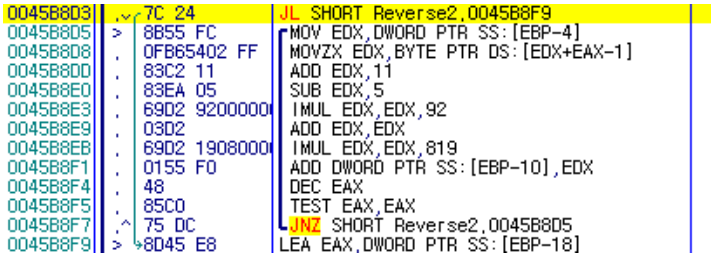


제가 입력한 Name값인 1234를 레지스터에 저장시킨 후 함수를 호출합니다.
저 함수가 제일 먼저 수상하여 분석해 보기로 하였습니다.



함수를 분석하다 다음과 같은 루틴을 볼수가 있었습니다.

저 루틴을 돌며 확인 한 결과 제가 입력한 1234의 값들을 차례대로 이용해 연산을 해주는 루틴이었습니다.
저 루틴의 결과로 EDX에 저장되는 값은 A672E340인것을 확인 할 수 있었습니다.
루틴의 결과값으로 나온 A672E340이 왠지 눈에 익어보여 1234를 입력할때의 key값과 비교해보니
앞 숫자 4자리가 key값의 맨 첫번째 부분이 되는 값이었습니다.
일단 여기서 "저 루틴에의해 key값의 첫번째 부분이 생성된다" 라는 가설을 만들고 다음 루틴을 진행해 보았습니다.



얼마 안가서 바로 두번째 반복문이 보이는데, 분석을 해보니 여기서는 거꾸로 1234를 거꾸로 이용하여 연산을 하는 루틴이었습니다. 이 루틴의 결과로 90518E8이 나왔는데 이 값 역시 키값의 두번째 부분의 키값인 9051과 관련이 있어보입니다. 여기서 또 "이 루틴에 의해 key값의 두번째 부분이 생성된다"라는 가설을 세웠습니다.

그리고 이 루틴들을 알아보기 쉽게 메모장에 정리했습니다.

```
5D88-53B4-52A87D27-1D0D-5B09

1234-> A672-9051-FE90F705-8EA9-BBD4
```

```
name[i]+0
name[i] *= 0x772
tmp = name[i] * name[i]
name[i] = tmp + name[i]
name[i] = name[i] * 0x474
name[i] += name[i]

name[3-i] + 0x11
name[3-i] - 0x05
name[3-i] *= 0x92
name[3-i] += name[3-i]
name[3-i] *= 0x819
sum += name[3-i]
```

이제 이것과 가설을 이용해서 코딩하여 name값을 찾아내는 일 밖에 안남았습니다.
만약 가설이 맞다면 뒤의 루틴은 관계없이 저것만으로도 Name값을 찾아 낼 수 있을겁니다.

```
>>> for i in pattern:
        for j in pattern:
            tmp = 0
            name = i+j
            for k in range(0,2):
                digit = ord(name[k])
                digit = digit + tmp
                digit = digit * 0x772
                tmp = digit * digit
                digit = digit + tmp
                digit = digit * 0x474
                digit += digit
                tmp = digit
            tmp = hex(tmp).upper()
            if tmp[len(tmp)-9:len(tmp)-5] == "5D88":
                print name
```

C6

0-9A-Za-z 패턴을 만든 후 차례대로 대입하여 루틴의 결과로 5D88이 나오는 값을 찾아보니
C6이라는 결과값을 얻었습니다.

C6이 맞나 확인하기위해 두번째 루틴을 프로그래밍해 결과값을 얻어보았습니다.

```
>>> for i in range(0,2):
        digit = ord(name[1-i])
        digit += 0x11
        digit -= 0x5
        digit *= 0x92
        digit += digit
        digit *= 0x819
        sum += digit


>>> hex(sum)
'0x53b46c4'
```

첫 숫자 4자리가 5D88-53B4-52A87D27-1D0D-5B09의 두번째값인 53B4와 일치하는걸 볼수가있었습니다 :D
가설은 맞아떨어졌네요 :D