CodeEngn Reverse2 L08

- 문제 -

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

힌트: Name은 두자리인데.. 알파벳일수도 있고 숫자일수도 있고.. 정답인증은 Name의 MD5 해쉬값(대문자)

- 풀이 -

프로그램을 수행하여 특징을 찾아본다. Name과 Key값을 다양하게 입력해 보니 Name을 2 글자 이하로 입력 했을때 Key의 텍스트박스에 "Please Enter More Chars..." 문구를 출력한다. 그이외 별다른 특징은 없어 보인다.

"Please Enter More Chars..." 문자를 통해 입력값을 검사하는 부분을 찾아보았다.

0045BB24	> 83F8 03	CMP EAX,3	
		JGE SHORT 47152C5A.0045BB3E	
0045BB29	. BA 18BC4500	MOV EDX,47152C5A.0045BC18	ASCII "Please Enter More Chars"
0045BB2E		MOV EAX, DWORD PTR DS: [EBX+374]	
0045BB34		CALL 47152C5A.0043A0A4	
0045BB39	. E9 91000000	JMP 47152C5A.0045BBCF	
0045BB3E	> >8D55 F4	LEA EDX.DWORD PTR SS: [EBP-C]	
0045BB41	. 8883 6803000	(MOV EAX,DWORD PTR DS:[EBX+368]	
0045BB47	. E8 28E5FDFF	CALL 47152C5A.0043A074	
0045BB4C	. 8845 F4	MOV EAX,DWORD PTR SS:[EBP-C]	
0045BB4F	. 8945 F8	MOV DWORD PTR SS:[EBP-8], EAX	
0045BB52	. 8845 F8	MOV EAX, DWORD PTR SS: [EBP-8]	
0045BB55	. 85C0	TEST EAX, EAX	
0045BB57	.⊍ 74 05	JE SHORT 47152C5A.0045BB5E	
0045BB59	. 83E8 04	SUB EAX,4	
0045BB5C		MOV EAX, DWORD PTR DS: [EAX]	
	> 83F8 1E	CMP EAX,1E	
0045BB61		JLE SHORT 47152C5A.0045BB75	
	. BA 3CBC4500	MOV EDX,47152C5A.0045BC3C	ASCII "Please Enter Not More Then 30 Chars"
0045BB68		MOV EAX, DWORD PTR DS: [EBX+374]	
	. E8 31E5FDFF	CALL 47152C5A.0043A0A4	
	.∨ EB 5A	JMP SHORT 47152C5A.0045BBCF	
	> 8D55 F0	LEA EDX,DWORD PTR SS: [EBP-10]	
0045BB78		MOV EAX,DWORD PTR DS:[EBX+374]	
	. E8 F1E4FDFF	CALL 47152C5A.0043A074	
	. 8845 F0	MOV EAX, DWORD PTR SS: [EBP-10]	
	. 50	PUSH EAX	
0045BB87	. 8D55 E8	LEA EDX,DWORD PTR SS: [EBP-18]	
0045BB8A		MOV EAX, DWORD PTR DS: [EBX+368]	
	. E8 DFE4FDFF	CALL 47152C5A.0043A074	
	. 8845 E8	MOV EAX, DWORD PTR SS: [EBP-18]	
	. 8D55 EC	LEA EDX,DWORD PTR SS:[EBP-14]	
0045BB9B		CALL 47152C5A.0045B850	
0045BBA0	5.0	MOV EDX,DWORD PTR SS:[EBP-14]	
0045BBA3 0045BBA4	. 58 . E8 9390FAFF	POP EAX CALL 47152C5A.00404C3C	
0045BBA4	. 75 1A	JNZ SHORT 47152C5A.00404C3C	
0045BBAB	. 6A 40	PUSH 40	
0045BBAD	. B9 64BC4500	MOV ECX,47152C5A.0045BC64	ASCII "Good Boy!!!"
0045BBB2	. BA 70BC4500	MOV ECX,47152C5A.0045BC64	ASCII "Well done!"
00430002	1 BA 706C4300	MOV EDA, 47132C3A, 0043BC70	ASCII WETT GOTE:

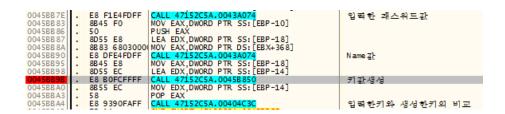
0045BB24에서 길이 값을 비교한 후 분기하는 것을 확인할 수 있다. 중간에 보니 30글자이상 입력을 하면 경고를 발생하는듯하다.

하단에 "Good Boy", "Well done" 문구가 보이는 것을 보니 0045BBAB에서 패스워드를 확인하여 분기한다는 것을 알 수 있다.

입력받은 값을 통해 키를 만들고 비교하는 구문은 0045BB7E에서 0045BBA4사이의 CALL 문에서 이루어진다는 것을 추측할 수 있다.

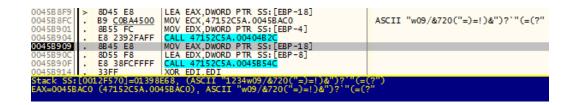
스택과 레지스터의 변화를 관찰하면서 함수 내부를 확인해본 결과 0043A074 함수에서는 입력한 패스워드값, 0043A074에서는 Name 값을 스택에 푸시 하였다.

0045B850 함수에서는 Name값을 통해 패스워드를 생성하였고, 00404C3C에서는 앞서 입력한 패스워드와 Name값을 통해 생성된 패스워드를 비교한후 0045BBA9에서 분기하였다.



Name에는 1234를 Key는 1111111111을 입력하였는데 0045BB9B함수를 지난후 스택에 보이는 킷값을 확인하니 "A672-9051-FE90F705-8EA9-BBD4" 이었다.

우선 주어진 문제를 해결하기 위해서 0045BB98주소에 브포를 걸고 함수를 탐색하였다.

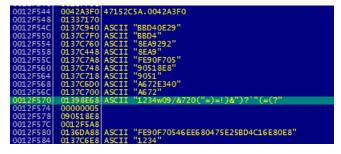


함수 내부의 0045B904 CALL문을 거치면서 입력한 Name값에 특정문자들을 붙이는 작업을 한다.

0045B90F에서 해쉬로 추정되는 값을 생성하였다.

한스탭씩 내려가며 스택 값의 변화를 살펴보면서 아래 그림과 같은 결과를 얻었다. 즉, 킷값에 해당하는 문자들이 각각의 CALL문을 지나면서 하나씩 생성되는 것을 확인하였다.

좀 더 자세히 분석하여 킷값이 생성되는 원리를 찾아본다.



[그림 4] 변화된 스택값

```
E8 05CDFAFF
8B45 E0
B9 04000000
BA 01000000
E8 3793FAFF
FF75 E4
68 E4BA4500
8D45 DC
0045B9BE
0045B9C3
                                                                        CALL 47152C5A.004086C8
MOV EAX,DWORD PTR SS:[EBP-20]
MOV ECX,4
                                                                                                                                                                                          첫번째 키문자열생성
 0045B9C6
 0045B9CB
0045B9D0
0045B9D5
                                                                        MOV EDX,1
                                                                       MOV EDX,1
CALL 47152C5A.00404D0C
PUSH DWORD PTR SS:[EBP-1C]
PUSH 47152C5A.0045BAE4
LEA EAX,DWORD PTR SS:[EBP-24]
PUSH EAX
LEA ECX,DWORD PTR SS:[EBP-28]
XOR EDX, EDX
MOV EAX,DWORD PTR SS:[EBP-10]
CALL 47152C5A.004086C8
                                                                                                                                                                                          위 문자열의 앞 4글자만 추출
 0045B9DD
 0045R9F0
                                  50
8D4D D8
33D2
8B45 F0
E8 DACCFAFF
 0045B9E6
                                                                        CALL 47152C5A.004086C8

MOV EAX,DWORD PTR SS:[EBP-28]

MOV ECX,4

MOV EDX,1

CALL 47152C5A.00404D0C
                                  E8 DACCFAFF
8845 D8
9 04000000
BA 01000000
E8 0C93FAFF
FF75 DC
68 E48A4500
8D45 D4
50
B9 08000000
BA 01000000
8B45 F8
E8 EE92FAFF
FF75 D4
68 E48A4500
8D45 D0
50
 0045B9E9
                                                                                                                                                                                          두번째 키문자열 생성
 0045B9EE
 004589F1
004589F6
004589F8
                                                                      MOV EDX,1
CALL 47152C5A.00404D0C
PUSH DWORD PTR SS: [EBP-24]
PUSH 47152C5A.0045BAE4
LEA EAX,DWORD PTR SS: [EBP-2C]
PUSH EAX
MOV ECX,8
MOV EOX,1
MOV EAX,DWORD PTR SS: [EBP-8]
CALL 47152C5A.00404D0C
PUSH DWORD PTR SS: [EBP-2C]
PUSH 47152C5A.0045BAE4
LEA EAX,DWORD PTR SS: [EBP-30]
PUSH EAX
                                                                                                                                                                                          위 문자열의 앞 4글자만 추출
 0045BA00
 0045BA03
 0045BA08
0045BA0B
0045BA0C
 0045BA11
 0045BA16
 0045BA19
0045BA1E
0045BA21
                                                                                                                                                                                          세번째 키문자열 생성
 0045BA26
                                                                        PUSH EAX
LEA ECX,DWORD PTR SS:[EBP-34]
XOR EDX,EDX
MOV EAX,EDI
 0045BA29
                                    8D4D CC
 0045BA2
                                  8D4D CC
33D2
8BC7
E8 92CCFAFF
8B45 CC
B9 04000000
BA 01000000
E8 C492FAFF
FF75 D0
                                                                       CALL 47152C5A.004086C8
MOV EAX,DWORD PTR SS:[EBP-34]
MOV ECX,4
MOV EDX,1
CALL 47152C5A.00404D0C
                                                                                                                                                                                          네번째 키 문자열 생성
 0045BA31
 0045BA36
 0045BA39
 0045BA3E
0045BA43
0045BA48
                                                                     MOV EDX,1
CALL 47152C5A.00404D0C
PUSH DWORD PTR SS:[EBP-30]
PUSH 47152C5A.0045BAE4
LEA EAX,DWORD PTR SS:[EBP-38]
PUSH EAX
LEA ECX,DWORD PTR SS:[EBP-3C]
XOR EDX,EDX
MOV EAX,EBX
CALL 47152C5A.004086C8
MOV EAX,DWORD PTR SS:[EBP-3C]
MOV ECX,4
MOV EDX,1
CALL 47152C5A.00404D0C
                                                                                                                                                                                          네번째 키 문자열 앞 4글자만 추출
                                   68 <u>E4BA4500</u>
8D45 C8
 0045BA4B
 0045BA50
                                   50
8D4D C4
33D2
 0045BA5
0045BA5
                                    8BC3
 0045BA59
0045BA5B
                                  E8 68CCFAFF
8B45 C4
B9 04000000
BA 01000000
E8 9A92FAFF
                                                                                                                                                                                          다섯번째 키 문자열 생성
 0045BA60
0045BA63
0045BA68
                                                                                                                                                                                          다섯번째 키 문자열 앞 4글자만 추훋
```

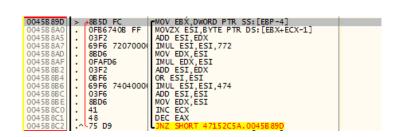
[그림 5] 킷값을 생성한 함수 내부

[그림 5]를 보면 004086C8 함수에서 킷값이 추출되는 것을 확인할 수 있다.

자세히 살펴보니 0045B9BE에서 호출하는 함수의 내부에서 PUSH하는 ESI값에 해당하는 문자열을 생성하였다. 즉, 0045B9BC의 MOV에서 ESI를 EAX에 복사한뒤 EAX값을 문자로 변환하는 작업을 진행하였다.

004086C8함수는 EAX값을 문자로 변환하는 함수임을 알 수 있다.

킷값을 생성하는 구문 위로 ESI값에 변화를 주는 구문을 OPCODE 확인을 하며 위로 추적해본 결과 다음과 같았다.



ESI를 생성하는 원리를 통해 주어진 문제의 첫 번째 킷값에 해당하는 문자를 알아내면 문제를 해결할 수 있을 것으로 보인다.

C코드로 구현하여 약간의 브루트포스를 통해 정답을 도출하였다.

```
#include <stdio.h>
#include <string.h>
int main()
          int edx = 0;
          int esi, eax;
          char Name[10]="11";
          int result=0x5D88;
          char str_code[]=
                    "0123456789 abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ";\\
          int i, j, k;
          int bEnd=1;
          for(j=0;j<strlen(str_code)&&bEnd;j++)</pre>
                     for(k=0;k<strlen(str_code)&&bEnd;k++)</pre>
                               Name[0]=str_code[k];
                               Name[1]=str_code[j];
                               edx=0;
                               for(i=0;i<strlen(Name);i++)
                                          esi = Name[i];
                                          esi += edx;
                                          esi *=0x772;
                                          edx = esi;
                                          edx *= esi;
                                          esi +=edx;
                                          esi |=esi;
                                          esi *=0x474;
                                          esi +=esi;
                                          edx = esi;
                               if((esi >> 16) == result)
                                          printf("%s\n",Name);
                                          bEnd=0;
                               }
          return 0;
```

결과로 C6의 값을 얻어올 수 있었다. C6을 MD5변환하여 결과를 도출하였다.

정답: 7E8B9F5CAB4A8FE24FAD9FE4B7452702