

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 2글자 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	
0045BB27	>	7D 15	JGE SHORT 47152C5A.0045BB3E	
0045BB29	.	BA 188C4500	MOV EDX,47152C5A.0045BC18	ASCII "Please Enter More Chars..."
0045BB2E	.	8883 74030000	MOV EAX,DWORD PTR DS:[EBX+374]	
0045BB34	.	E8 68E5FDFF	CALL 47152C5A.0043A0A4	
0045BB39	.	E9 91000000	JMP 47152C5A.0045BBCF	
0045BB3E	>	8D55 F4	LEA EDX,DWORD PTR SS:[EBP-C]	
0045BB41	.	8883 68030000	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	.	8800	MOV EAX,DWORD PTR DS:[EAX]	
0045BB5E	>	83F8 1E	CMP EAX,1E	
0045BB61	>	7E 12	JLE SHORT 47152C5A.0045BB75	
0045BB63	.	BA 3C8C4500	MOV EDX,47152C5A.0045BC3C	ASCII "Please Enter Not More Than 30 Chars..."
0045BB68	.	8883 74030000	MOV EAX,DWORD PTR DS:[EBX+374]	
0045BB6E	.	E8 31E5FDFF	CALL 47152C5A.0043A0A4	
0045BB73	>	EB 5A	JMP SHORT 47152C5A.0045BBCF	
0045BB75	>	8D55 F0	LEA EDX,DWORD PTR SS:[EBP-10]	
0045BB78	.	8883 74030000	MOV EAX,DWORD PTR DS:[EBX+374]	
0045BB7E	.	E8 F1E4FDFF	CALL 47152C5A.0043A074	
0045BB83	.	8845 F0	MOV EAX,DWORD PTR SS:[EBP-10]	
0045BB86	.	50	PUSH EAX	
0045BB87	.	8D55 E8	LEA EDX,DWORD PTR SS:[EBP-18]	
0045BB8A	.	8883 68030000	MOV EAX,DWORD PTR DS:[EBX+368]	
0045BB90	.	E8 DFE4FDFF	CALL 47152C5A.0043A074	
0045BB95	.	8845 E8	MOV EAX,DWORD PTR SS:[EBP-18]	
0045BB98	.	8D55 EC	LEA EDX,DWORD PTR SS:[EBP-14]	
0045BB9B	.	E8 B0FCFFFF	CALL 47152C5A.0045B850	
0045BBA0	.	8855 EC	MOV EDX,DWORD PTR SS:[EBP-14]	
0045BBA3	.	58	POP EAX	
0045BBA4	.	E8 9390FAFF	CALL 47152C5A.00404C3C	
0045BBA9	>	75 1A	JNZ SHORT 47152C5A.0045BB85	
0045BBAB	.	6A 40	PUSH 40	
0045BBAD	.	B9 648C4500	MOV ECX,47152C5A.0045BC64	ASCII "Good Boy!!!"
0045BBB2	.	BA 708C4500	MOV EDX,47152C5A.0045BC70	ASCII "Well done!"

0045BB24에서 길이 값을 비교한 후 분기하는 것을 확인할 수 있다. 중간에 보니 30글자 이상 입력을 하면 경고를 발생하듯하다.

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.

0045B87E	. E8 F1E4DFFF	CALL 47152C5A.0043A074	입력한 패스워드값
0045B883	. 8845 F0	MOV EAX,DWORD PTR SS:[EBP-10]	
0045B886	. 50	PUSH EAX	
0045B887	. 8055 E8	LEA EDX,DWORD PTR SS:[EBP-18]	
0045B88A	. 8883 68030000	MOV EAX,DWORD PTR DS:[EBX+368]	
0045B890	. E8 DFE4DFFF	CALL 47152C5A.0043A074	Name값
0045B895	. 8845 E8	MOV EAX,DWORD PTR SS:[EBP-18]	
0045B898	. 8055 EC	LEA EDX,DWORD PTR SS:[EBP-14]	
0045B89B	. E8 80FCFFFF	CALL 47152C5A.0045B850	킷값생성
0045B8A0	. 8855 EC	MOV EDX,DWORD PTR SS:[EBP-14]	
0045B8A3	. 58	POP EAX	
0045B8A4	. E8 9390FAFF	CALL 47152C5A.00404C3C	입력한키와 생성한키의 비교

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.

0045B8F9	> 8D45 E8	LEA EAX,DWORD PTR SS:[EBP-18]	
0045B8FC	. B9 C0BA4500	MOV ECX,47152C5A.0045BAC0	ASCII "w09/&720(="=!)&")?"'="(?"
0045B901	. 8855 FC	MOV EDX,DWORD PTR SS:[EBP-4]	
0045B904	. E8 2392FAFF	CALL 47152C5A.0040482C	
0045B909	. 8845 E8	MOV EAX,DWORD PTR SS:[EBP-18]	
0045B90C	. 8D55 F8	LEA EDX,DWORD PTR SS:[EBP-8]	
0045B90F	. E8 38FCFFFF	CALL 47152C5A.0045B54C	
0045B914	. 33FF	XOR EDI,EDI	
Stack SS:[0012F570]=01398E68, (ASCII "1234w09/&720(="=!)&")?"'="(?" EAX=0045BAC0 (47152C5A.0045BAC0), ASCII "w09/&720(="=!)&")?"'="(?"			

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

0045B90F에서 해쉬로 추정되는 값을 생성하였다.

한스텝씩 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.

0012F544	0042A3F0	47152C5A.0042A3F0
0012F548	01337170	
0012F54C	0137C940	ASCII "8BD40E29"
0012F550	0137C7F0	ASCII "8BD4"
0012F554	0137C760	ASCII "8EA9292"
0012F558	0137C448	ASCII "8EA9"
0012F55C	0137C7A8	ASCII "FE90F705"
0012F560	0137C748	ASCII "90518E8"
0012F564	0137C718	ASCII "9051"
0012F568	0137C6D0	ASCII "A672E340"
0012F56C	0137C700	ASCII "A672"
0012F570	01398E68	ASCII "1234w09/&720(="=!)&")?"'="(?"
0012F574	00000005	
0012F578	090518E8	
0012F57C	0012F5A8	
0012F580	0136DA88	ASCII "FE90F70546EE680475E25BD4C16E80E8"
0012F584	0137C6E8	ASCII "1234"

[그림 4] 변화된 스택값

0045B9BE	E8 05CDFAFF	CALL 47152C5A.004086C8	첫번째 키문자열 생성
0045B9C3	8B45 E0	MOV EAX,DWORD PTR SS:[EBP-20]	
0045B9C6	B9 04000000	MOV ECX,4	
0045B9CB	BA 01000000	MOV EDX,1	
0045B9D0	E8 3793FAFF	CALL 47152C5A.00404D0C	위 문자열의 앞 4글자만 추출
0045B9D5	FF75 E4	PUSH DWORD PTR SS:[EBP-1C]	
0045B9D8	68 E4BA4500	PUSH 47152C5A.0045BAE4	
0045B9DD	8D45 DC	LEA EAX,DWORD PTR SS:[EBP-24]	
0045B9E0	50	PUSH EAX	
0045B9E1	8D4D D8	LEA ECX,DWORD PTR SS:[EBP-28]	
0045B9E4	33D2	XOR EDX,EDX	
0045B9E6	8B45 F0	MOV EAX,DWORD PTR SS:[EBP-10]	
0045B9E9	E8 DACCF AFF	CALL 47152C5A.004086C8	두번째 키문자열 생성
0045B9EE	8B45 D8	MOV EAX,DWORD PTR SS:[EBP-28]	
0045B9F1	B9 04000000	MOV ECX,4	
0045B9F6	BA 01000000	MOV EDX,1	
0045B9FB	E8 0C93FAFF	CALL 47152C5A.00404D0C	위 문자열의 앞 4글자만 추출
0045BA00	FF75 DC	PUSH DWORD PTR SS:[EBP-24]	
0045BA03	68 E4BA4500	PUSH 47152C5A.0045BAE4	
0045BA08	8D45 D4	LEA EAX,DWORD PTR SS:[EBP-2C]	
0045BA0B	50	PUSH EAX	
0045BA0C	B9 08000000	MOV ECX,8	
0045BA11	BA 01000000	MOV EDX,1	
0045BA16	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]	
0045BA19	E8 EE92FAFF	CALL 47152C5A.00404D0C	세번째 키문자열 생성
0045BA1E	FF75 D4	PUSH DWORD PTR SS:[EBP-2C]	
0045BA21	68 E4BA4500	PUSH 47152C5A.0045BAE4	
0045BA26	8D45 D0	LEA EAX,DWORD PTR SS:[EBP-30]	
0045BA29	50	PUSH EAX	
0045BA2A	8D4D CC	LEA ECX,DWORD PTR SS:[EBP-34]	
0045BA2D	33D2	XOR EDX,EDX	
0045BA2F	8BC7	MOV EAX,EDI	
0045BA31	E8 92CCFAFF	CALL 47152C5A.004086C8	네번째 키 문자열 생성
0045BA36	8B45 CC	MOV EAX,DWORD PTR SS:[EBP-34]	
0045BA39	B9 04000000	MOV ECX,4	
0045BA3E	BA 01000000	MOV EDX,1	
0045BA43	E8 C492FAFF	CALL 47152C5A.00404D0C	네번째 키 문자열 앞 4글자만 추출
0045BA48	FF75 D0	PUSH DWORD PTR SS:[EBP-30]	
0045BA4B	68 E4BA4500	PUSH 47152C5A.0045BAE4	
0045BA50	8D45 C8	LEA EAX,DWORD PTR SS:[EBP-38]	
0045BA53	50	PUSH EAX	
0045BA54	8D4D C4	LEA ECX,DWORD PTR SS:[EBP-3C]	
0045BA57	33D2	XOR EDX,EDX	
0045BA59	8BC3	MOV EAX,EBX	
0045BA5B	E8 68CCFAFF	CALL 47152C5A.004086C8	다섯번째 키 문자열 생성
0045BA60	8B45 C4	MOV EAX,DWORD PTR SS:[EBP-3C]	
0045BA63	B9 04000000	MOV ECX,4	
0045BA68	BA 01000000	MOV EDX,1	
0045BA6D	E8 9A92FAFF	CALL 47152C5A.00404D0C	다섯번째 키 문자열 앞 4글자만 추출
0045BA73	FF75 C8	PUSH DWORD PTR SS:[EBP-38]	

[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 OP CODE 확인을 하며 위로 추적 해본 결과 다음과 같았다.

0045B89D	> 8B5D FC	MOV EBX,DWORD PTR SS:[EBP-4]
0045B8A0	0FB67408 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 47152C5A.0045B89D

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[]=
        "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";

    int i, j, k;
    int bEnd=1;

    for(j=0;j<strlen(str_code)&&bEnd;j++)
    {
        for(k=0;k<strlen(str_code)&&bEnd;k++)
        {
            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