Reversing.kr Easy Keygen 2 풀이

리버싱 문제풀이 / Wonlf / 2022. 3. 30. 17:40



압축 해제를 하고 보니, ReadMe.txt가 있다 읽어보겠다.

ReversingKr KeygenMe

Find the Name when the Serial is 5B134977135E7D13

Serial이 특정 문자열일 때, Name을 찾아 달라고 한다.

먼저, Die에 넣어보았다.



패킹이 되어있지 않고, 32비트 프로그램이다.

파일을 실행시켜본다.

```
□ C:#Users#a02*#Downloads#Easy_KeygenMe#Easy Keygen.exe — X
Input Name: whatiskey
Input Serial: serial

A
```

ReadMe에서 봤던 serial은 여기 부분을 말하는 것 같다. 디버깅을 시작해보겠다.

```
81EC 30010000
00401000 | [$
                                                                                             sub_401000
                                           sub esp,130
00401006
               55
                                           push ebp
00401007
               56
                                           push esi
00401008
               57
                                           push edi
               B9 18000000
00401009
                                           mov ecx, 18
               33C0
                                           xor eax,eax
0040100E
00401010
               8D7C24 11
                                           lea edi,dword ptr ss:[esp+11]
                                          mov byte ptr ss:[esp+10],0
mov byte ptr ss:[esp+74],0
rep stosd
               C64424 10 00
C64424 74 00
00401014
00401019
0040101E
               F3:AB
00401020
               66:AB
                                           stosw
00401022
                                           stosb
00401023
               B9 31000000
                                           mov ecx,31
                                                                                             31: '1'
00401028
               33C0
                                           xor eax,eax
               8D7C24 75
                                           lea edi,dword ptr ss:[esp+75]
0040102A
                                                                                             408060: "Input Name: "
               68 60804000
0040102E
                                           push easy keygen.408060
00401033
                                           rep stosá
               F3:AB
00401035
               66:AB
                                           stosw
00401037
                                           stosb
                                          mov byte ptr ss: esp+10],10
mov byte ptr ss: esp+11],20
mov byte ptr ss: esp+12],30
00401038
               C64424 10 10
                                                                                             20:''
30:'0'
0040103D
               C64424 11 20
               C64424 12 30
00401042
                                           call <easy keygen.sub_4011B9>
add esp,4
00401047
               E8 6D010000
               83C4 04
0040104C
               8D4424 10
0040104F
                                           lea eax,dword ptr ss:[esp+10]
00401053
               50
                                           push eax
                                          push easy keygen.40805C
call <easy keygen.sub_4011A2>
lea edi,dword ptr ss:[esp+18]
                                                                                             40805C: "%s"
00401054
               68 5C804000
00401059
               E8 44010000
0040105E
               8D7C24 18
               83C9 FF
00401062
                                           or ecx, FFFFFFF
00401065
               33C0
                                           xor eax, eax
00401067
               83C4 08
                                           add esp.8
0040106A
                                           xor ebp, ebp
                33ED
0040106C
               33F6
                                          xor esi,esi
```

00401000 함수에서 문자열을 출력하고 입력 받는 구문을 발견하였다. Name을 입력해주고 아래 구문으로 한줄씩 넘어가보면,

```
00401073
00401075
                85C9
                                         test ecx,ecx
                                          jle easy keygen.4010B6
                7E 3F
00401077
                83FE 03
                                         cmp esi,3
                                         jl easy keygen.40107E
xor esi,esi
0040107A
                7C 02
                33F6
                                         movsx ecx,byte ptr ss:[esp+esi+C]
movsx edx,byte ptr ss:[esp+ebp+10]
● 0040107E
                OFBE4C34 OC
                0FBE542C 10
00401083
00401088
                33CA
                                         xor ecx,edx
                8D4424 74
0040108A
                                         lea eax,dword ptr ss:[esp+74]
● 0040108E
                                         push ecx
                                         push eax
lea ecx,dword ptr ss:[esp+7C]
● 0040108F
00401090
                8D4C24 7C
                68 54804000
                                         push easy keygen.408054
00401094
                                                                                        408054: "%s%02X"
00401099
                51
                                         push ecx
0040109A
                E8 B1000000
                                          call <easy keygen.sub_401150>
● 0040109F
                83C4 10
                                         add esp,10
  004010A2
                                         inc ebp
                                         lea edi,dword ptr ss:[esp+10]
or ecx,FFFFFFFF
● 004010A3
                8D7C24 10
                                                                                         [esp+10]:EntryPoint
중단점 설정되지 않음
                83C9 FF
               33C0
                                         xor eax, eax
● 004010AC
                46
                                         inc esi
● 004010AD
                F2:AE
                                         repne scasb
● 004010AF
                F7D1
                                         not ecx
● 004010B1
                49
                                         dec ecx
                                         cmp ebp,ecx
jl easy keygen.401077
● 004010B2
                3BE9
                7C C1
B9 19000000
004010B4
004010B6
                                         mov ecx,19
● 004010BB
                33C0
                                         xor eax,eax
```

for문 loop안에 들어오게 되고 [esp+esi+C]안의 값을 가져오고 [esp+ebp+10]의 값을 가져온다. 여기 안에 있는 값들을 살펴보면,

```
0019FDF8 EA 11 40 00 EA 11 40 00 70 FF 19 00 10 20 30 00 ê.@.ê.@.pÿ.. 0. 0019FE08 77 68 61 74 69 73 68 65 79 00 00 00 00 00 00 00 whatiskey.....
```

[esp+esi+C] 안에는 0x10

[esp+ebp+10] 안에는 내가 입력한 값의 첫번째 값 이렇게 들어있다.

더 아래로 가면 내가 입력한 값의 첫번째 값과 0x10을 xor하고 이 값을 sub_401150 함수에서 호출하며 인자값으로 넣는다.

그리고 메모리를 보게되면

```
0 ê.@.ê.@.pÿ... 0.
0 whatiskey.....
0 .....
0 .....
0 .....
0 .....
0 ....
0 ....
0 ....
0 ....
0 ....
0 ....
```

67이라는 16진수가 **문자열**로 메모리에 저장 되었다.

그리고 두번째 loop를 돌게 되면,

[esp+esi+C] 안에는 0x20

[esp+ebp+10] 안에는 내가 입력한 값의 두번째 값 이렇게 들어있다.

또 아래 구문에서 내가 입력한

값의 두번째 값과 0x20을 xor하여 메모리에 저장한다.

계속 이런식으로 확인해보면,

- 1. 내가 입력한 값의 첫번째 값과 0x10을 xor하여 저장한다.
- 2. 내가 입력한 값의 두번째 값과 0x20을 xor하여 저장한다.
- 3. 내가 입력한 값의 세번째 값과 0x30을 xor하여 저장한다.
- 4. 내가 입력한 값의 네번째 값과 0x10을 xor하여 저장한다.

여기서 중요한점은 0x10 ~ 0x30까지 xor을 하고 4번째 값부터는 다시 0x10으로 돌아와 xor을 한다는 점과 2자리 문자열로 저장한다는 점이다.

Name의 값을 가공하는 과정을 알아냈으니, 더 아래로 내려가본다.

시리얼을 입력하고 바로 아래 구문을 보게 되면,

```
8D7424 74
                                      lea esi,dword ptr ss:[esp+74
               8D4424 10
                                      lea eax, dword ptr ss: esp+10
                                                                                 eax:"serial"
   004010EA
               8A10
                                      mov dl,byte ptr ds:[eax]
               8ACA
                                      mov cl,dl
> 004010EE
                                      cmp dl,byte ptr ds:[esi]
                                                                                 esi:"6748516449437B4549"
                3A16
                                          easy keygen.40110E
                75
                  1C
● 004010F2
               84C9
                                      test cl,cl
● 004010F4
                                       je easy keygen.40110A
               74 14
                                                                                 eax+1:"erial"
● 004010F6
               8A50 01
                                      mov d],byte ptr ds:[eax+1]
● 004010F9
                                      mov cl,dl
               8ACA
                3A56 01
                                                                                 esi+1: "748516449437B4549"
● 004010FB
                                      cmp dl,byte ptr ds:[esi+1]
- 004010FE
               75 OE
                                      jne easy keygen.40110E
```

내가 입력한 값의 첫번째 값을 특정 값과 xor 한 것을 가져와 serial과 비교하는 구문이 있다.

여기서 정리해볼 수 있다. 나온다.

입력한 값의 자리 값과 특정 패턴에 따른 수를 xor한 값은 16진수로 나오게 되고 2자리 문자열로 메모리에 저장된다.

- 1. 내가 입력한 값의 1번째 값과 0x10을 xor하면 0x5B가 나온다.
- 2. 내가 입력한 값의 2번째 값과 0x20을 xor하여 0x13이 나온다.

- 3. 내가 입력한 값의 3번째 값과 0x30을 xor하여 0x49가 나온다.
- 4. 내가 입력한 값의 4번째 값과 0x10을 xor하여 0x77이 나온다.
- 5. 내가 입력한 값의 5번째 값과 0x20을 xor하여 0x13이 나온다
- 6. 내가 입력한 값의 6번째 값과 0x30을 xor하여 0x5E가 나온다.
- 7. 내가 입력한 값의 7번째 값과 0x10을 xor하여 0x7D가 나온다.
- 8. 내가 입력한 값의 8번째 값과 0x20을 xor하여 0x13이 나온다.

name이 처음에 xor해서 나온 16진수를 문자열로 저장 했으니, 역산 할때는 16진수로 해주어야된다. 이를 토대로 코드를 짜보면,

```
#include <stdio.h>
int main() {
    int serial[] = \{0x5B, 0x13, 0x49, 0x77, 0x13, 0x5E, 0x7D, 0x13\};
    int value[3] = \{0x10, 0x20, 0x30\};
    int count = 0;
    for (int i = 0; i < sizeof(serial) / sizeof(int); i++) {</pre>
        if(count == 2){
            printf("%c", serial[i] ^ value[count]);
            count = 0;
        }
        else{
            printf("%c", serial[i] ^ value[count]);
            count++;
        }
    }
    return 0;
}
```

정답은 "K3yg3nm3"