# 攻防世界 reverse 的logmein writeup

原创

上山砍大树 于 2019-11-14 17:01:33 发布 644 收藏 3

分类专栏： 逆向 文章标签： 攻防世界reserve





```
void __fastcall __noreturn main(__int64 a1, char **a2, char **a3)
{
  size_t v3; // rsi
  int i; // [rsp+3Ch] [rbp-54h]
  char input_string[36]; // [rsp+40h] [rbp-50h]
  int v6; // [rsp+64h] [rbp-2Ch]
  __int64 v7; // [rsp+68h] [rbp-28h]
  char v8[8]; // [rsp+70h] [rbp-20h]
  int v9; // [rsp+8Ch] [rbp-4h]

  v9 = 0;
  strcpy(v8, ":\"AL_RT^L*.?+6/46");
  v7 = 0x65626D61726168LL;
  v6 = 7;
  printf("Welcome to the RC3 secure password guesser.\n", a2, a3);
  printf("To continue, you must enter the correct password.\n");
  printf("Enter your guess: ");
  __isoc99_scanf("%32s", input_string);
  v3 = strlen(input_string);
  if ( v3 < strlen(v8) )                        // 输入字符串长度>=strlen（v8）
    quit();
  for ( i = 0; i < strlen(input_string); ++i )
  {
    if ( i >= strlen(v8) )                      // 从这里就可以判断出输入字符串长度=strlen(v8)
      quit();
    if ( input_string[i] != (char)(*((_BYTE *)&v7 + i % v6) ^ v8[i]) )// (&v7+i%v6)==>地址+偏移，就是数组了。
                                                // 指针+偏移里就是数组的寻参方式
      quit();
  }
  sub_4007F0();
}
```



main函数的反汇编如上图。

拿到ida静态分析了一下，通读后main的反汇编后的代码，大体了解了flag应该是通过`(char)(*((_BYTE *)&v7 + i % v6) ^ v8[i])`

来确定的。

然后就是一些小细节了

## 1.input_string的长度

```
if ( v3 < strlen(v8) ) //strlen(input_string)>=strlen(v8)
  quit();
if ( i >= strlen(v8) )//strlen(input_string)<=strlen(v8)
      quit();
```

所以得出`strlen(input_string)==strlen(v8)`
然后我们看一下如何确定输入

## 2.&v7

```
input_string[i] != (char)(*((_BYTE *)&v7 + i % v6) ^ v8[i])
```

v7是long long int类型，占8byte=64bit，故int64来定义v7。
我们一层层刨析这段代码：
（_BYTE*）&v7;//就是强制v7转化为char类型数据，使得&v7是一个字符串的首地址，如果不好理解，可以用另一种表达方式来表示相同的意思：

```
char *target=(char *)&v7;//即用一个char类型指针指向转化为char类型后的v7的首地址
```

那么v7地址保存的数据是什么呢？那就需要了解数据在内存中是以字节为单位存储然后在intelx86的兼容机下以小端序保存数据。
用一张图来解释上面的那句话：
注：*long long int v7=2853719457361956O转化为16进制后为：0x65626D61726168*；

| 偏移地址 | 数据 | ASCII |
|---|---|---|
| 01 | 0x68 | h |
| 02 | 0x61 | a |
| 03 | 0x72 | r |
| 04 | 0x61 | a |
| 05 | 0x6d | m |
| 06 | 0x62 | b |
| 07 | 0x65 | e |

然后(target+i)=(&v7+i)，相当于用指针索引数据，地址按字节编址，在c中指针加1指的是增加一个存储一个单元，对于数组而言，这意味着加1后的地址是下一个元素的地址，而不是下一个字节的地址。（可以好好体味c primer plus中关于指针和数组的分析）

所以（char*)v7=“harambe”;

```
((_BYTE *)&v7 + i % v6)//意思就是(char *)v7[i%v6]
```

然后了解后直接写一个c程序来得到最终正确的输入：

```
/*第一个程序*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    long long int v7=0x65626D61726168;
    char a[]=":\"AL_RT^L*.?+6/46";
    char input_string[20];

    for(int i=0;i<strlen(a);i++)
    {
        input_string[i]=(char)(*((char *)&v7 + i%7)^a[i]);
        printf("%c",input_string[i]);
    }
    printf("\n\n%d",sizeof(long long int));
    return 0;
}
```

也可以用指针来指向v7来索引数据：

```
#include <stdio.h>
#include <string.h>

int main()
{
    char v8[]=":\"AL_RT^L*.?+6/46";//strlen(v8)==strlen(input_string)
    long long int v7=0x65626D61726168;

    int i;
    /*将long long int转化为字符类型*/
    char *target=(char *)&v7;

    char final_string[20];
    for(i=0;i<strlen(v8);i++)
    {
        final_string[i]=target[i%7]^v8[i];
        printf("%c",final_string[i]);
    }
    return 0;

}
```

最终的结果都是`RC3-2016-XORISGUD`
这个题用到了指针和数组的联系，小端序字节存储的问题，不明白的可以再体会一下。希望大家共同进步吧！