jactf source writeup

原创

qq_43710556 于 2019-07-26 21:44:18 发布 69 收藏

分类专栏： CTF

本文链接：https://blog.csdn.net/qq_43710556/article/details/97418591

CTF 专栏收录该内容

5 篇文章 0 订阅

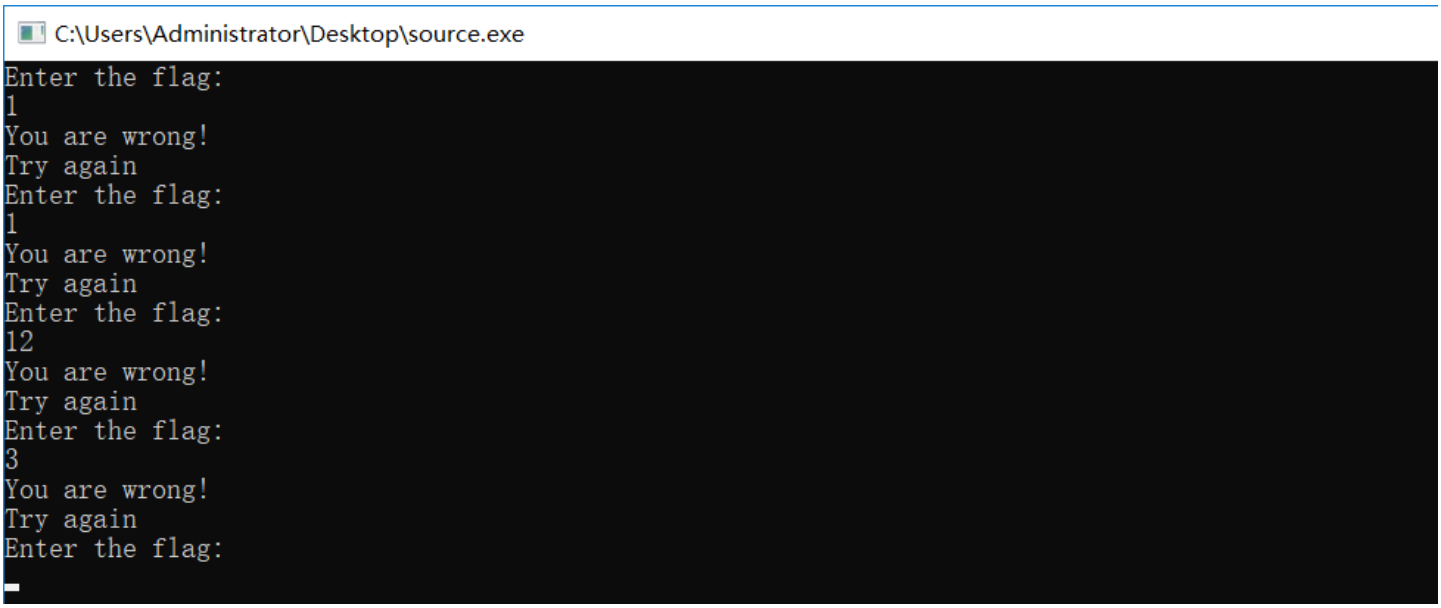
订阅专栏

jactf source writeup

今天在jactf做了一道逆向，对逆向题目又有了新的理解，特此记录。

下载完题目之后首先查壳

无壳，接着打开看看这个程序的内容（习惯）。

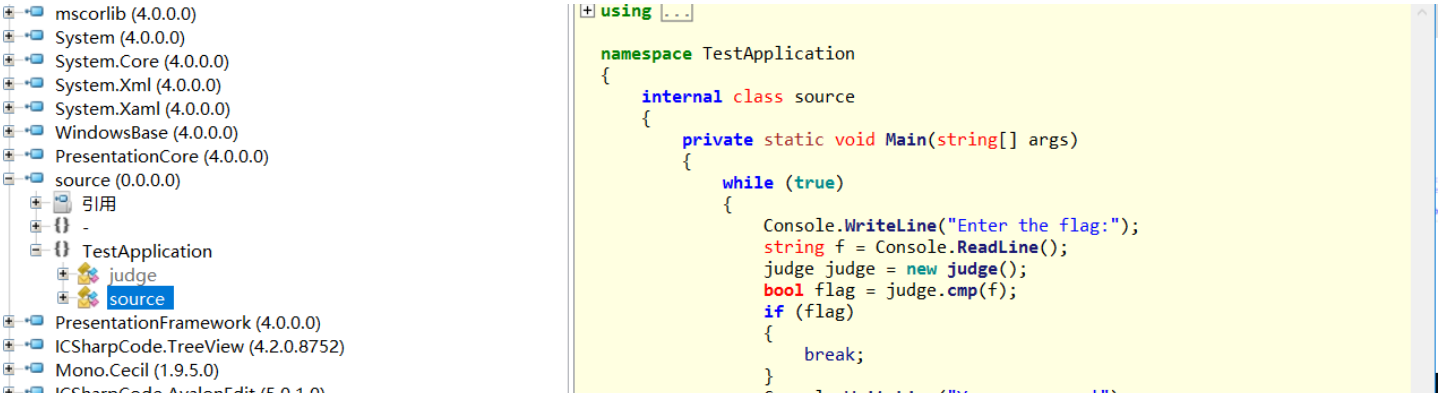


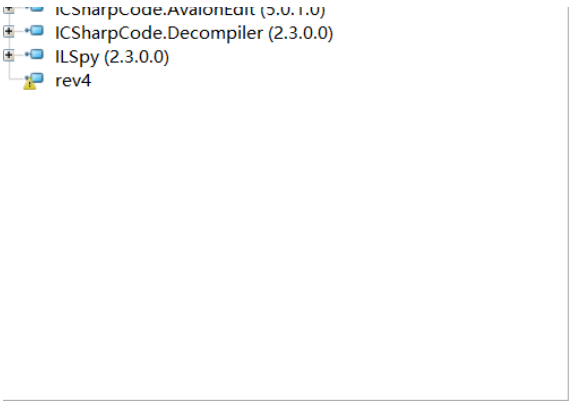
发现让输入flag。
这种题目也做过好几道了，先放到IDA里面看看。

发现这啥也没有啊。。
放到kali里面查看一下文件类型。

```
root@kali:~# file '/root/桌面/source'
/root/桌面/source: PE32 executable (console) Intel 80386 Mono/.Net
embly, for MS Windows
```

是.net文件(抱住小小的自己)。。。
不怕，还好之前了解过.net文件的逆向。
.net 文件使用ILSpy工具进行逆向分析。
其实ILSpy工具的使用很简单，感觉不需要专门的学习，摸索一会儿也就了解了。
好吧，继续开始逆向。
用ILSpy打开source文件






```
                Console.WriteLine("You are wrong!");
                Console.WriteLine("Try again");
            }
            Console.WriteLine("You are right!");
            Console.ReadKey();
        }
    }
}
```


发现程序也不复杂，由一个主函数和一个judge判断函数组成。

先看主函数。

十分明了，一眼就能看出来程序是干什么的。

然后再看看judge函数。

```
using ...

namespace TestApplication
{
    internal class judge
    {
        private string KEY1 = "flag{Thi3_i3+A_wrong+str}";

        private int[] KEY2 = new int[]
        {
            24,
            90,
            51,
            23,
            66,
            172,
            49,
            34,
            246,
            240,
            25,
            27,
            224,
            88,
            253,
            50,
            254,
            10,
            7,
            31,
            84,
            5,
            12,
            38,
            15,
```



```
        public int[] encrypt(string key, int seed, string strings)
        {
            int length = strings.Length;
            int[] array = new int[29];
            int[] array2 = new int[25];
            int[] array3 = new int[29];
            byte[] array4 = new byte[1];
            byte[] array5 = new byte[1];
```

```
            array4 = Encoding.ASCII.GetBytes(strings);
            array5 = Encoding.ASCII.GetBytes(key);
            for (int i = 0; i < length; i++)
            {
                array3[i] = (int)array4[i];
            }
            for (int j = 0; j < 25; j++)
            {
                array2[j] = (int)array5[j];
            }
            for (int k = 0; k < length; k++)
            {
                array[k] = (array3[k] + seed ^ array2[seed]) % 255;
                seed = (seed + 1) % 25;
            }
            return array;
        }

        public bool cmp(string F)
        {
            int[] array = this.encrypt(this.KEY1, 7, F);
            bool result = true;
            for (int i = 0; i < 29; i++)
            {
                if (this.KEY2[i] != array[i])
                {
                    result = false;
```


一眼看去judge函数很长（仔细分析后其实也不难）
首先定义了一个KEY1="flag{Thi3_i3+A_wrong+str}"的字符串。
（发现jactf的题总是拿假的flag骗人。。让萌新以为这就完事了呢）
然后又定义了一个KEY2的数组。
之后就是encrypt的加密函数。先不分析，继续向下看。

```
    public bool cmp(string F)
    {
        int[] array = this.encrypt(this.KEY1, 7, F);
        bool result = true;
        for (int i = 0; i < 29; i++)
        {
            if (this.KEY2[i] != array[i])
            {
                result = false;
                break;
            }
        }
        return result;
    }
}
```


这个应该是个cmp函数调用了encrypt函数。
将KEY1,7,F作为参数传入encrypt函数，KEY1的值我们已经知道了，F的值就是我们要输入的值，
所以到这里我们的任务已经很明确了，就是逆向推导出正确的F。
array就是经过加密的数组了。
在向下就是将array的每一位和KEY2进行比较。
所以我们可以知道，加密后的array==KEY2
我们再回去分析encrypt函数。

```
public int[] encrypt(string key, int seed, string strings)
{
```

```
    int length = strings.Length;
    int[] array = new int[29];
    int[] array2 = new int[25];
    int[] array3 = new int[29];
    byte[] array4 = new byte[1];
    byte[] array5 = new byte[1];
    array4 = Encoding.ASCII.GetBytes(strings);
    array5 = Encoding.ASCII.GetBytes(key);
    for (int i = 0; i < length; i++)
    {
        array3[i] = (int)array4[i];
    }
    for (int j = 0; j < 25; j++)
    {
        array2[j] = (int)array5[j];
    }
    for (int k = 0; k < length; k++)
    {
        array[k] = (array3[k] + seed ^ array2[seed]) % 255;
        seed = (seed + 1) % 25;
    }
    return array;
}
```


定义了 3个数组，2个字符串。。

array4就是传入的F,也就是我们需要输入的字符串。这里划重点。如果能求出array4的值，我们也就求出了F的值。

array5就是传入的KEY1

所以array5=“flag{Thi3_i3+A_wrong+str}”

再看第一个循环。

就是将array4字符串的每一位转化为整数再付给array3

真是一波三折啊。。我们现在的目标就是求出array3的值。。

第二个循环同上。。

第三个循环就是重点了。。

这个length。。。

做的时候分析了一下，就是F 的长度。而F的长度和array3的长度是相同的。。

看上面array3定义的时候定义的长度是29所以这个length就是29（下面cmp函数也证实了）

array[k]=(array3[k]+seed^array2[seed])%255

这个表达式，其实除了array3的值我们都已经知道了。
到这里就明了了。
可以开始写脚本了。
写脚本的时候这个%取余的运算真是难倒我了（数学没学好，心疼自己一秒）
写了半个多小时，还没搞定这个%取余运算。。
突然想到，arrray3对应的是字符串，那它的范围就在（0,255）之间。
所以换个思路，爆破array3
所以最终的脚本就是：

```
key1='flag{Thi3_i3+A_wrong+str}'
key2=[24,90,51,23,66,172,49,34,246,240,25,27,224,88,253,50,254,10,7,31,84,5,12,38,15,16,79,117,238]
array5=key1
array2=[]
array3=[]
for i in range(0,25):
array2.append(ord(array5[i]))
array=key2
seed=7
for i in range(0,29):
for j in range(0,256):
if array[i]==(j+seed^array2[seed])%255:
array3.append(j)
break
seed=(seed+1)%25
for i in array3:
print(chr(i),end='')
```