2019 GXY_Re部分_WriteUp

原创

Hk_Mayfly 于 2019-12-27 15:02:00 发布 168 收藏 1

赛题下载地址：**https://www.lanzous.com/b07r91skd**

0x01 lucky_guy

IDA打开

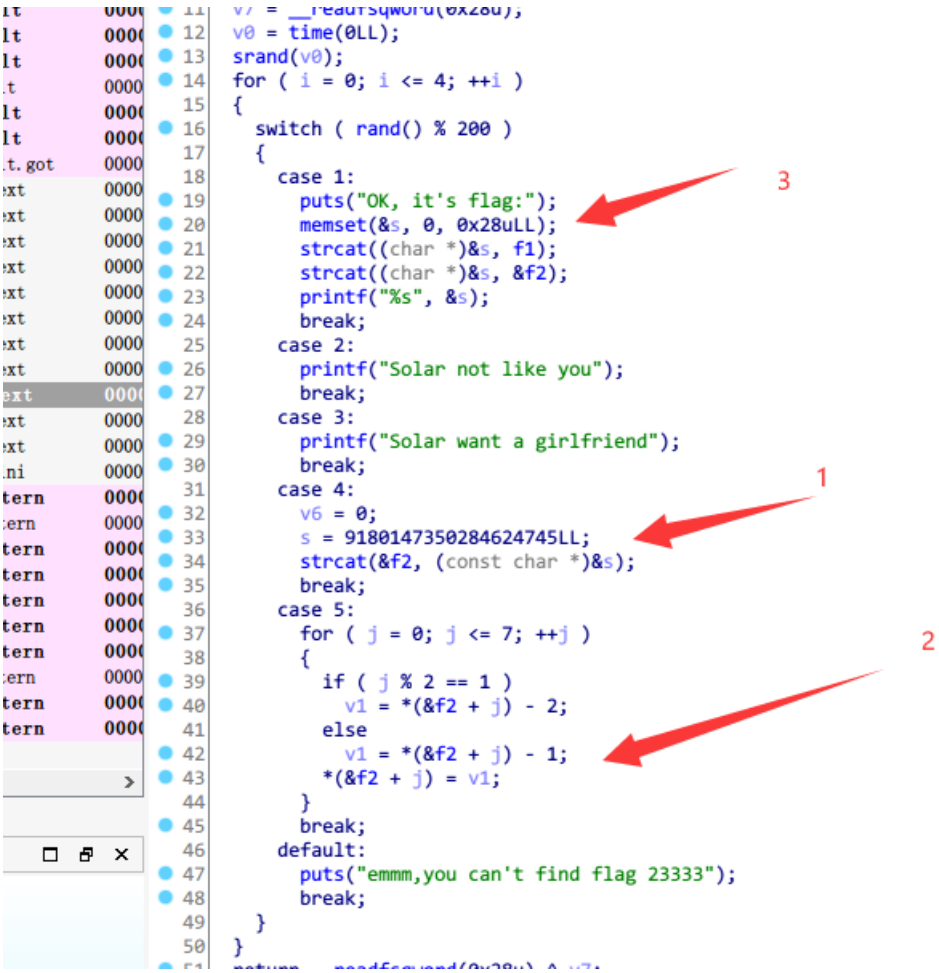
进入patch_me

进入get_flag

随机产生5个数字，每产生一个进入下面switch进行处理

代码分析

分析得到处理顺序




```
12  v0 = time(0LL);
13  srand(v0);
14  for ( i = 0; i <= 4; ++i )
15  {
16    switch ( rand() % 200 )
17    {
18      case 1:
19        puts("OK, it's flag:");
20        memset(&s, 0, 0x28uLL);
21        strcat((char *)&s, f1);
22        strcat((char *)&s, &f2);
23        printf("%s", &s);
24        break;
25      case 2:
26        printf("Solar not like you");
27        break;
28      case 3:
29        printf("Solar want a girlfriend");
30        break;
31      case 4:
32        v6 = 0;
33        s = 9180147350284624745LL;
34        strcat(&f2, (const char *)&s);
35        break;
36      case 5:
37        for ( j = 0; j <= 7; ++j )
38        {
39          if ( j % 2 == 1 )
40            v1 = *(&f2 + j) - 2;
41          else
42            v1 = *(&f2 + j) - 1;
43          *(&f2 + j) = v1;
44        }
45        break;
46      default:
47        puts("emmm,you can't find flag 23333");
48        break;
49    }
50  }
```

1中得到f2的初始值"icug`of "，2可能执行多次，3中f1为 "GXY{do_not_"，因此可以写出C++代码模拟过程。

程序解密

```
#include <iostream>

using namespace std;
/* run this program using the console pauser or add your own getch, system("pause") or input loop */

int main(int argc, char** argv) {
    char v1;
    char f2[] = "icug`of ";
    cout << f2 << endl;
    for (int i = 0; i < 4; ++i) {
        for (int j = 0; j <= 7; ++j) {
            if (j % 2 == 1)
                v1 = f2[j] - 2;
            else
                v1 = f2[j] - 1;
            f2[j] = v1;
        }
        cout << f2 << endl;
    }

    return 0;
}
```

得到后半部分，"hate_me}"

组合起来就是GXY{do_not_hate_me}

get flag！

> **GXY{do_not_hate_me}**

0x02 simple_CPP

IDA打开

将代码整体分为了三个部分

```
if ( v37 > 0 )
  {
    v5 = 0i64;
    do
    {
      v6 = &Memory;                             // v6为输入
      if ( v38 >= 0x10 )
        v6 = Memory;
      v7 = &Dst;                                // Dst=i_will_check_is_debug_or_not
      if ( (unsigned __int64)qword_140006060 >= 0x10 )
        v7 = (void **)Dst;
      v3[v5] = v6[v5] ^ *((_BYTE *)v7 + v4++ % 27);// 输入进行变换
      ++v5;
    }
    while ( v4 < v37 );
  }
```

```
  do
  {
    v14 = *v13 + v8;                            // 对输入变换后的字符串再进行变换
    ++v12;
    ++v13;
    switch ( v12 )
    {
      case 8:                                   // v11=第8位的
        v11 = v14;
        goto LABEL_24;
      case 16:
        v10 = v14;                              // v11第16位的
        goto LABEL_24;
      case 24:                                  // v11=第24位
        v9 = v14;
LABEL_24:
        v14 = 0i64;
        break;
      case 32:
        sub_1400019C0(std::cout, "ERRO,out of range");
        exit(1);
        break;
    }
    v8 = v14 << 8;
  }
  while ( v12 < (signed int)v37 );
```

```
 1  v35 = v15[2];                                 // 1-2
 2  v16 = v15[1];
 3  v17 = *v15;                                   // 1-2
 4  v18 = sub_14000223C(0x20ui64);
 5  if ( IsDebuggerPresent() )
 6  {
 7    sub_1400019C0(std::cout, "Hi , DO not debug me !");
 8    Sleep(0x7D0u);
 9    exit(0);
10  }
11  v19 = v16 & v17;
12  *v18 = v16 & v17;
13  v20 = v35 & ~v17;                             // 1-1
14  v18[1] = v20;
15  v21 = ~v16;
16  v22 = v35 & v21;
17  v18[2] = v35 & v21;
18  v23 = v17 & v21;
19  v18[3] = v23;
20  if ( v20 != 1176889593874i64 )
21  {
22    v18[1] = 0i64;
23    v20 = 0i64;
24  }
25  v24 = v20 | v19 | v22 | v23;
26  v25 = v15[1];
27  v26 = v15[2];
28  v27 = v22 & *v15 | v26 & (v19 | v25 & ~*v15 | ~(v25 | *v15));
29  v28 = 0;
30  if ( v27 == 577031497978884115i64 )
31    v28 = v24 == 4483974544037412639i64;
32  if ( (v24 ^ v15[3]) == 448397454319547011li64 )
33    v0 = v28;
34  if ( (v20 | v19 | v25 & v26) != (~*v15 & v26 | 8646933325792000l2i64) || v0 != 1 )
35  {
36    sub_1400019C0(std::cout, "Wrong answer!try again");
37    j_j_free(v3);
38  }
39  else
40  {
41    v29 = sub_1400019C0(std::cout, "Congratulations!flag is GXY{");
42    v30 = &Memory;
43    if ( v38 >= 0x10 )
44      v30 = Memory;
45    v31 = sub_140001FD0(v29, v30, v37);
46    sub_1400019C0(v31, "}");
47    j_j_free(v3);
48  }
```

代码分析

针对第三部分：

首先整理第三部分的各个运算式子

v19 = v15[1] & v17[0]
v20 = v15[2] & ~v15[0]
v20 = 1176889593874
v21 = ~v15[1]
v22 = v15[2] & ~v15[1]
v23 = v15[0] & ~v15[1]
v24 = (v15[2] & ~v15[0]) | (v15[1] & v15[0]) | (v15[2] & ~v15[1]) | (v15[0] & ~v15[1])
v24 = 4483974544037412639
v25 = v15[1]
v26 = v15[2]
v27 = (v15[2] & ~v15[1]) & v15[0] | v15[2] & ((v15[1] & v17[0]) | v15[1] &~v15[0] | ~(v15[1] | v15[0]))
v27 = 577031497978884115
v24 ^ v15[3] = 4483974543195470111
((v15[2] & ~v15[0]) | (v15[1] & v17[0]) | v15[1] & v15[2]) == (~v15[0] & v15[2] | 864693332579200012)

使用z3解方程，得到v15[0]~v15[3]的解

```
# -*- coding:utf-8 -*-

from z3 import *

x,y,z,w=BitVecs('x y z w',64)

s=Solver()

s.add((~x)&z==1176889593874)
s.add(((z&~x)|(x&y)|(z&(~y))|(x&(~y)))^w==4483974543195470111)
s.add(((z&~y)&x|z&((x&y)|y&~x|~(y|x)))==577031497978884115)
s.add(((z&~x)|(x&y)|(z&~y)|(x&~y))==4483974544037412639)
s.add(((z&(~x)) | (x&y) | y & z) == (((~x)& z)|864693332579200012))

s.check()
m = s.model()
for i in m:
    print("%s = 0x%x"%(i,m[i].as_long()))
```

w = 0x32310600
z = 0x8020717153e3013
y = 0xc00020130082c0c
x = 0x3e3a460533286f0d

针对第二部分：

将x,y,z,w拼接起来，并将x,y,z补足为16位

```
li=[]
# 拼接
li.append(hex(m[x].as_long())[2:].rjust(16,"0"))
li.append(hex(m[y].as_long())[2:].rjust(16,"0"))
li.append(hex(m[z].as_long())[2:].rjust(16,"0"))
li.append(hex(m[w].as_long())[2:-2])
print(li)
```

> ['3e3a460533286f0d', '0c00020130082c0c', '08020717153e3013', '323106']

针对第一部分：

就是一个异或的表达式，通过动态调试可以得到Dst的值为"i_will_check_is_debug_or_not"，即可得到v7，再将异或逆向就行。

脚本解密

```
# -*- coding:utf-8 -*-

from z3 import *

debug_str = "i_will_check_is_debug_or_not"
x,y,z,w=BitVecs('x y z w',64)

s=Solver()

s.add((~x)&z==1176889593874)
s.add(((z&~x)|(x&y)|(z&(~y))|(x&(~y)))^w==4483974543195470111)
s.add(((z&~y)&x|z&((x&y)|y&~x|~(y|x)))==577031497978884115)
s.add(((z&~x)|(x&y)|(z&~y)|(x&~y))==4483974544037412639)
s.add(((z&(~x)) | (x&y) | y & z) == (((~x)& z)|864693332579200012))

s.check()
m = s.model()
for i in m:
    print("%s = 0x%x"%(i,m[i].as_long()))
flag=""
li=[]
# 拼接
li.append(hex(m[x].as_long())[2:].rjust(16,"0"))
li.append(hex(m[y].as_long())[2:].rjust(16,"0"))
li.append(hex(m[z].as_long())[2:].rjust(16,"0"))
li.append(hex(m[w].as_long())[2:-2])
print(li)

v4=0
for i in li:
    for j in range(0,len(i),2):
        xx = i[j]+i[j+1]
        flag += chr(int(xx,16)^ord(debug_str[(v4)%27]))
        v4 += 1
    print(flag)
```

解出来得到的flag为

> **We1l_D0n**
> **We1l_D0ndeajoa_S**
> **We1l_D0ndeajoa_Slgebra_a**
> **We1l_D0ndeajoa_Slgebra_am_i**

get flag!

通过官方给的提示，第二部分有问题，给了第二部分的解“e!P0or_a”，替换掉我们得到的，得到正确的flag

> **GXY{We1l_D0ne!P0or_algebra_am_i}**

完整WP：链接1，链接2