第18天：红帽杯2019-childRE

原创

S1lenc3 于 2019-11-17 19:29:53 发布 479 收藏 1
分类专栏： CTF

本文链接：https://blog.csdn.net/qq_41858371/article/details/103111366

继续复现红帽杯逆向题。

程序流程：

```
sub_7FF7650B1080("%s", &v13);
v0 = -1i64;
do
  ++v0;
while ( *((_BYTE *)&v13 + v0) );
if ( v0 != 31 )
{
  while ( 1 )
    Sleep(0x3E8u);
}
```

输入并且检测长度，可以看到长度是31。

```
v1 = sub_7FF7650B1280(&v13);
v2 = name;
if ( v1 )
{
  sub_7FF7650B15C0((unsigned __int8 *)v1[1]);
  sub_7FF7650B15C0(*(unsigned __int8 **)(v3 + 16));
  v4 = dword_7FF7650B57E0;
  v2[v4] = *v5;
  dword_7FF7650B57E0 = v4 + 1;
}
```


这是一组变换函数，搞了好长时间才弄懂。

```
UnDecorateSymbolName(v2, outputString, 0x100u, 0);
v6 = -1i64;
do
  ++v6;
while ( outputString[v6] );
if ( v6 == 62 )
{
  v9 = 0;
  v10 = 0i64;
  do
```

对v2进行反修饰，得出的结果存入outputString，outputString长度为62.

```
do
{
  v11 = outputString[v10];
  v12 = v11 % 23;
  if ( a1234567890Qwer[v12] != *(_BYTE *)(v10 + 0x7FF7650B3478i64) )
    _exit(v9);
  if ( a1234567890Qwer[v11 / 23] != *(_BYTE *)(v10 + 0x7FF7650B3438i64) )
    _exit(v9 * v9);
  ++v9;
  ++v10;
}
while ( v9 < 0x3E );
sub_7FF7650B1020("flag{MD5(your input)}\n", v11 / 23, v12, v10);
result = 0i64;
```


取outputString的字符串进行除和取余，并且作为a1234567890Qwer的索引，对指定索引进行明文比较。

这就是正确的程序流程。

解题思路：

因为最后的比较都是明文，所以可以写脚本得到正确的outputString。

```
a1 = '1234567890-=!@#$%^&*()_+qwertyuiop[]QWERTYUIOP{}asdfghjkl;\'ASDFGHJKL:"ZXCVBNM<>?zxcvbnm,./'
b2 = '5556565325555222556556555524346633465366354442656555525555222'
b1 = '(_@4620!08!6_0*0442!@186%%0@3=66!!974*3234=&0^3&1@=&0908!6_0*&'
ops = ''
for i in range(len(b1)):
    index1 = a1.index(b2[i])
    index2 = a1.index(b1[i])
    ops += chr(index1 * 23 + index2)
print(ops)
```

然后对outputString进行函数名修饰，这里给一个大佬的思路，C++代码：

```
#include<iostream>
using namespace std;

class ROPxx {
public:
 ROPxx(){
  unsigned char a;
  My_Aut0_PWN(&a);
 }

private:
 char My_Aut0_PWN(unsigned char*) {
  printf("%s", __FUNCDNAME__);
  return '0';
 }
};
int main() {
 new ROPxx();
 getchar();
 return 0;
}
```

运行就能得到修饰结果。这样就拿到了正确的v2.

对v2进行逆变换得到输入的字符串，然后进行md5加密，就是最终的flag。

```
str1 = '?My_Aut0_PWN@R0Pxx@@AAEPADPAE@Z'
result = [''] * 31
index3 = [15, 16, 7, 17, 18, 8, 3, 19, 20, 9, 21, 22, 10, 4, 1, 23, 24, 11, 25, 26, 12, 5, 27, 28, 13, 29,
print(len(str1))
for i in range(len(str1)):
    result[index3[i]] += str1[i]
ss = ''.join(i for i in result)
print(ss)
print(hashlib.md5(ss.encode('utf-8')).hexdigest())
```

复现问题

这道题当时比赛没看，复现的时候，发现明文比较，反修饰这些都能看懂。但是搞不懂对输入变换这部分。花了一点时间来解决这个问题。

```
}
v1 = sub_7FF7650B1280(&v13);
v2 = name;

{
  sub_7FF7650B15C0((unsigned __int8 *)v1[1]);
  sub_7FF7650B15C0(*(unsigned __int8 **)(v3 + 16));
  v4 = dword_7FF7650B57E0;
  v2[v4] = *v5;
  dword_7FF7650B57E0 = v4 + 1;
```

两个点，

v1那个函数是什么用，

另外两个相同的函数又是干什么的。

网上的writeup都直接说动态调试发现是固定的位置变换，直接把索引提取出来了。思路好一致。。。。

看了官方writeup才知道：

v1后边那个函数是对输入建立一个二叉树，31个字符，是五层满二叉树。

另外两个相同的函数是后序遍历，

调试了一下`v1 = sub_7FF7650B1280(&v13);`这个函数，

怎么识别是二叉树呢？

```
v2 = sub_7FF7650B1D44(0x18ui64);
*(_BYTE *)v2 = 0;
v2[1] = 0i64;
v2[2] = 0i64;
```

动态调试会发现，开辟的地址处有三个数据，第一个是我们输入的字符，第二个是左孩子的地址，第三个是右孩子的地址。

可以识别出是二叉树结构体。

```
sub_7FF7650B15C0((unsigned __int8 *)v1[1]);
sub_7FF7650B15C0(*(unsigned __int8 **)(v3 + 16));
```

这是进行后序遍历。

第一个是对左半边进行遍历，

第二个是对右半边进行遍历。

```
  if ( a1 )
  {
    v1 = a1;
    sub_7FF7650B15C0(*((unsigned __int8 **)a1 + 1));
    sub_7FF7650B15C0(*((unsigned __int8 **)v1 + 2));
    v2 = dword_7FF7650B57E0;
    result = *v1;
    name[v2] = result;
    dword_7FF7650B57E0 = v2 + 1;
  }
  return result;
}
```


函数内进行递归。

找出索引如下，因为是从0开始，所以需要减一。