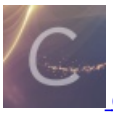
2021[HFCTF]虎符杯网络安全赛道re部分wp

原创

N1ptune__ 于 2021-04-07 19:25:06 发布 822 收藏 1

分类专栏： ctf 逆向 文章标签： c++ python c语言 安全

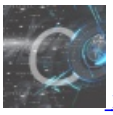
本文链接：https://blog.csdn.net/xshower/article/details/115493179



ctf 同时被 2 个专栏收录

2 篇文章 0 订阅

订阅专栏



逆向

4 篇文章 0 订阅

订阅专栏

CrackMe

这题是使用c++写的，因为无符号表，ida对c++的解析不是很友好，可以去了解一下c++的string类型的结构，对解题很有帮助，推荐看这篇文章链接: https://bbs.pediy.com/thread-230312.htm.

```
    }
    else
    {
      v104 = v4 + 1;                                // v11=input[0:7]
      v11 = v103;
      if ( v5 >= 0x10 )
        v11 = (void **)v103[0];
      *((_BYTE *)v11 + v4) = v10;
      *((_BYTE *)v11 + v4 + 1) = 0;
    }
    if ( ++v8 >= 7 )
      break;
    v5 = v105;
    v4 = v104;
  }
  v12 = 7i64;
  while ( 1 )
  {
    v13 = input1;
    if ( max_len >= 0x10 )
      v13 = (void **)input1[0];
    v14 = *((_BYTE *)v13 + v12);
    if ( v6 >= v7 )
    {                                               // v15=input[7:17]
      sub_7FF6F5D12750(v106, v5, v3, v14);
    }
    else
    {
      v107 = v6 + 1;
      v15 = v106;
      if ( v7 >= 0x10 )
        v15 = (void **)v106[0];
      *((_BYTE *)v15 + v6) = v14;
      *((_BYTE *)v15 + v6 + 1) = 0;
    }
```

程序先将长度为17位的输入分为7位和10位两部分

再输入一个数要求满足一定条件

```
92  std::istream::operator>>(std::cin, &v96);
93  v16 = 0.0;
94  v17 = 0.0;
95  v18 = 0.0;
96  v19 = (double)((int)v96 / 12379) + 1.0;
97  do
98  {
99    v17 = v17 + *(double *)sub_7FF6F5D11360(v18, v19).m128_u64 * 0.001;
00    v18 = v18 + 0.001;
01  }
02  while ( v18 <= 100.0 );
03  v20 = (int)(v17 + v17 + 3.0);
04  v21 = 0.0;
05  v22 = (double)((int)v96 % 12379) + 1.0;
06  do
07  {
08    v16 = v16 + *(double *)sub_7FF6F5D11360(v21, v22).m128_u64 * 0.001;
09    v21 = v21 + 0.001;
10  }
11  while ( v21 <= 100.0 );
12  if ( v20 == dword_7FF6F5D17044 && (int)(v16 + v16 + 3.0) == dword_7FF6F5D17048 )// 这两个数在别的地方被修改过
13                                                // 直接动调查看最终比较结果
14  {
```


直接爆破

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double ff(double a,double b){
    double c=pow(a,b-1);
    return c/exp(a);
}

int main()
{
    int i=0;
    while(1){

        double v17=0.0;
        double v18=0.0;
        double v19=(double)i+1.0;
        do{
            v17=v17+ff(v18,v19)*0.001;
            v18=v18+0.001;
        }while(v18<=100.0);
        double v20=(int)(v17+v17+3.0);
        if(v20==0x13B03){
            printf("success!!  %d\n",i);
            break;
        }
        else{
            printf("n0!!   %d\n",i);
        }
        i++;
    }
    int j=0;
    for(;j<12379;j++){
        double v16=0.0;
        double v21=0;
        double v22=(double)j+1.0;
        do{
            v16=v16+ff(v21,v22)*0.001;
            v21=v21+0.001;
        }while(v21<=100.0);
        if((int)(v16+v16+3.0)==0x5a2){
            printf("success!!  %d\n",j);
            break;
        }
        else{
            printf("n0!!   %d\n",j);
        }
    }

    printf("%d",i*12379+j);
    return 0;
}
```

最后得到data=99038

观察程序，发现我们的输入被分成两组后，在分别做一个异或，最后比较，第二组的加密程序似乎是一个RC4，但无所谓，直接找到异或的数值

```
    {
      v42 = v103;
      if ( v37 >= 0x10 )
        v42 = v38;
      v43 = (char *)v42 + v41 % v39;
      v44 = Block;
      if ( v26 >= 0x10 )
        v44 = v25;
      *((_BYTE *)v44 + v40) ^= *v43;          // 异或操作
      ++v36;
      ++v40;
      v41 = v36;
      v26 = v99;
      v25 = (void **)Block[0];
    }
    while ( v36 < v39 );
  }
  for ( i = 0i64; i < 7; ++i )
  {
    v46 = Block;
    if ( v26 >= 0x10 )
      v46 = v25;
    if ( *((_BYTE *)v46 + i) != *((_BYTE *)&v94 + i) )// 比较
    {
      if ( v30 >= 0x10 )
      {
        v86 = v31;
        if ( v30 + 1 >= 0x1000 )
        {
```



```
  {
    v71 = 0i64;
    do
    {
      v68 = (unsigned __int8)(v68 + 1);
      v72 = *(_DWORD *)&v49[4 * v68 + 8];
      v69 = (unsigned __int8)(v72 + v69);
      v73 = *(_DWORD *)&v49[4 * v69 + 8];
      *(_DWORD *)&v49[4 * v68 + 8] = v73;
      *(_DWORD *)&v49[4 * v69 + 8] = v72;
      *((_BYTE *)&v112 + v71++) ^= v49[4 * (unsigned __int8)(v72 + v73) + 8];// 异或
    }
    while ( v71 < v70 );
  }
  *(_DWORD *)v49 = v68;
  *((_DWORD *)v49 + 1) = v69;
  v94 = 0x545314AA3F8ED6B2i64;
  LOWORD(v95) = 0x6C6;
  v74 = 0;
  for ( j = 0i64; j < 10; ++j )
  {
    if ( *((_BYTE *)&v94 + j) != *((_BYTE *)&v112 + j) )// 比较
      exit(-v74);
    ++v74;
  }
  v76 = sub_7FF6F5D123B0(std::cout, "flag{");
  v77 = input1;
```


直接动调找到异或和比较的数据，写脚本解密

```
first=[0x8,0x4d,0x59,0x6,0x73,0x2,0x40]
key='9903819'
flag=''
for i in range(len(first)):
    flag+=chr(ord(key[i])^first[i])


second=[0xe0,0x95,0xba,0x60,0xc9,0x66,0x2a,0x24,0xb2,0x36]
key_pre=[0xb2,0xd6,0x8e,0x3f,0xaa,0x14,0x53,0x54,0xc6,0x6]

for i in range(len(second)):
    flag+=chr(second[i]^key_pre[i])
print(flag)
###1ti5K3yRC4_crypt0
```

RE(忘了名字了，第一道题)

mips程序

用ghidra或者ida都可以，个人习惯ida

关键函数

```
16  v3 = std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::length(a2);
17  v10 = v3;
18  if ( v3 >= 0x200000 )
19    _cxa_throw_bad_array_new_length();
20  s = (void *)operator new[](v3 << 10);
21  memset(s, 0, v10 << 10);
22  *((_DWORD *)s
23  + *(char *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](a2, 0)) = 1;
24  v4 = 0;
25  for ( i = 1; i < v10; ++i )
26  {
27    for ( j = 0; j < 256; ++j )
28    {
29      if ( j != *(char *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](a2, i) )
30        *((_DWORD *)s + 256 * i + j) = *((_DWORD *)s + 256 * v4 + j);
31      else
32        *((_DWORD *)s + 256 * i + j) = i + 1;
33    }
34    v4 = *((_DWORD *)s
35         + 256 * v4
36         + *(char *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](a2, i));
37  }
38  v7 = 0;
39  for ( k = 0; k < v9; ++k )
40  {
41    v7 = *((_DWORD *)s
42         + 256 * v7
43         + *(char *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](a1, k));
44    if ( v7 == v10 )
45      return k - v10 + 1;
46  }
47  return -1;
```


这里我简单推到后发现v4恒等于0？

满足结果时a1和a2的前14位相同，直接将前14位输入发现可以

flag{Ninja Must Die}