攻防世界 Reverse高手进阶区 2分题 crazy

原创

思源湖的鱼 于 2020-12-30 14:41:28 发布 223 收藏 1

分类专栏： ctf 文章标签： ctf 攻防世界 reverse

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/weixin_44604541/article/details/111987578

版权

ctf 专栏收录该内容

200 篇文章 23 订阅

订阅专栏

前言

继续ctf的旅程

攻防世界Reverse高手进阶区的2分题

本篇是crazy的writeup

发现攻防世界的题目分数是动态的

就仅以做题时的分数为准了

解题过程

PE查壳

扔进IDA

```
24
25  v24 = __readfsqword(0x28u);
26  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(&v18, argv, envp);
27  std::operator>><char,std::char_traits<char>,std::allocator<char>>(&std::cin, &v18);
28  v3 = std::operator<<<std::char_traits<char>>(&std::cout, "----------------------------------------");
29  std::ostream::operator<<(v3, &std::endl<char,std::char_traits<char>>);
30  v4 = std::operator<<<std::char_traits<char>>(&std::cout, "Quote from people's champ");
31  std::ostream::operator<<(v4, &std::endl<char,std::char_traits<char>>);
32  v5 = std::operator<<<std::char_traits<char>>(&std::cout, "----------------------------------------");
33  std::ostream::operator<<(v5, &std::endl<char,std::char_traits<char>>);
34  v6 = std::operator<<<std::char_traits<char>>(
35        &std::cout,
36        "*My goal was never to be the loudest or the craziest. It was to be the most entertaining.");
37  std::ostream::operator<<(v6, &std::endl<char,std::char_traits<char>>);
38  v7 = std::operator<<<std::char_traits<char>>(&std::cout, "*Wrestling was like stand-up comedy for me.");
39  std::ostream::operator<<(v7, &std::endl<char,std::char_traits<char>>);
40  v8 = std::operator<<<std::char_traits<char>>(
41        &std::cout,
42        "*I like to use the hard times in the past to motivate me today.");
43  std::ostream::operator<<(v8, &std::endl<char,std::char_traits<char>>);
44  v9 = std::operator<<<std::char_traits<char>>(&std::cout, "----------------------------------------");
45  std::ostream::operator<<(v9, &std::endl<char,std::char_traits<char>>);
46  HighTemplar::HighTemplar(&v23, &v18);
47  v10 = std::operator<<<std::char_traits<char>>(&std::cout, "Checking....");
48  std::ostream::operator<<(v10, &std::endl<char,std::char_traits<char>>);
49  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(&v19, &v18);
50  func1(&v20, &v19);
51  func2(&v21, &v20);
52  func3(&v21, 0LL);
53  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(&v21);
54  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(&v20);
55  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(&v19);
56  HighTemplar::calculate((HighTemplar *)&v23);
57  if ( (unsigned int)HighTemplar::getSerial((HighTemplar *)&v23) == 0 )
58  {
59    v11 = std::operator<<<std::char_traits<char>>(&std::cout, "////////////////////////////////");
60    std::ostream::operator<<(v11, &std::endl<char,std::char_traits<char>>);
61    v12 = std::operator<<<std::char_traits<char>>(&std::cout, "Do not be angry. Happy Hacking :)");
62    std::ostream::operator<<(v12, &std::endl<char,std::char_traits<char>>);
63    v13 = std::operator<<<std::char_traits<char>>(&std::cout, "////////////////////////////////");
64    std::ostream::operator<<(v13, &std::endl<char,std::char_traits<char>>);
65    ZN11HighTemplar7getFlagB5cxx11Ev(&v22, &v23);
66    v14 = std::operator<<<std::char_traits<char>>(&std::cout, "flag{");
67    v15 = std::operator<<<char,std::char_traits<char>,std::allocator<char>>(v14, &v22);
68    v16 = std::operator<<<std::char_traits<char>>(v15, "}");
69    std::ostream::operator<<(v16, &std::endl<char,std::char_traits<char>>);
70    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(&v22);
71  }
72  HighTemplar::~HighTemplar((HighTemplar *)&v23);
73  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(&v18);
74  return 0;
75}
```

https://blog.csdn.net/weixin_44604541

有点乱

依次找关键函数

```
1  unsigned __int64 __fastcall HighTemplar::HighTemplar(DarkTemplar *a1, __int64 a2)
2  {
3    char v3; // [rsp+17h] [rbp-19h]
4    unsigned __int64 v4; // [rsp+18h] [rbp-18h]
5
6    v4 = __readfsqword(0x28u);
7    DarkTemplar::DarkTemplar(a1);
8    *(_QWORD *)a1 = &off_401EA0;
9    *((_DWORD *)a1 + 3) = 0;
10   std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string((__int64)a1 + 16, a2);
11   std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string((__int64)a1 + 48, a2);
12   std::allocator<char>::allocator(&v3, a2);
13   std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(
14     (char *)a1 + 80,
15     "327a6c4304ad5938eaf0efb6cc3e53dc",
16     &v3);
17   std::allocator<char>::~allocator(&v3);
18   return __readfsqword(0x28u) ^ v4;
19 }
```

https://blog.csdn.net/weixin_44604541

主要是赋值

```
bool __fastcall HighTemplar::calculate(HighTemplar *this)
{
  __int64 v1; // rax
  _BYTE *v2; // rbx
  bool result; // al
  _BYTE *v4; // rbx
  int i; // [rsp+18h] [rbp-18h]
  int j; // [rsp+1Ch] [rbp-14h]

  if ( std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::length((char *)this + 16) != 32 )
  {
    v1 = std::operator<<<std::char_traits<char>>(&std::cout, "Too short or too long");
    std::ostream::operator<<(v1, &std::endl<char,std::char_traits<char>>);
    exit(-1);
  }
  for ( i = 0;
        i <= (unsigned __int64)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::length((char *)this + 16);
        ++i )
  {
    v2 = (_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](
                    (char *)this + 16,
                    i);
    *v2 = (*(_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](
                       (char *)this + 16,
                       i) ^ 0x50)
        + 23;
  }
  for ( j = 0; ; ++j )
  {
    result = j <= (unsigned __int64)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::length((char *)this + 16);
    if ( !result )
      break;
    v4 = (_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](
                    (char *)this + 16,
                    j);
    *v4 = (*(_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](
                       (char *)this + 16,
                       j) ^ 0x13)
        + 11;
  }
  return result;
}
```

https://blog.csdn.net/weixin_44604541

判断长度

```
__int64 __fastcall HighTemplar::getSerial(HighTemplar *this)
{
  __int64 v1; // rbx
  __int64 v2; // rax
  __int64 v3; // rax
  __int64 v4; // rax
  __int64 v5; // rax
  unsigned int i; // [rsp+1Ch] [rbp-14h]

  for ( i = 0;
        (signed int)i < (unsigned __int64)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::length((char *)this + 16);
        ++i )
  {
    v1 = *(unsigned __int8 *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](
                               (char *)this + 80,
                               (signed int)i);
    if ( (_BYTE)v1 != *(_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](
                                  (char *)this + 16,
                                  (signed int)i) )
    {
      v4 = std::operator<<<std::char_traits<char>>(&std::cout, "You did not pass ");
      v5 = std::ostream::operator<<(v4, i);
      std::ostream::operator<<(v5, &std::endl<char,std::char_traits<char>>);
      *((_DWORD *)this + 3) = 1;
      return *((unsigned int *)this + 3);
    }
    v2 = std::operator<<<std::char_traits<char>>(&std::cout, "Pass ");
    v3 = std::ostream::operator<<(v2, i);
    std::ostream::operator<<(v3, &std::endl<char,std::char_traits<char>>);
  }
  return *((unsigned int *)this + 3);
}
```

https://blog.csdn.net/weixin_44604541

验证

```
__int64 __fastcall ZN11HighTemplar7getFlagB5cxx11Ev(__int64 a1, __int64 a2)
{
  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(a1, a2 + 48);
  return a1;
}
```

发现关键是calculate函数

```
s='327a6c4304ad5938eaf0efb6cc3e53dc'
flag=''
for i in range(len(s)):
 flag+=chr((((ord(s[i])-11)^0x13)-23)^0x50)
print('flag{'+flag+'}')
```

得到flag：`flag{tMx~qdstOs~crvtwb~aOba}qddtbrtcd}`

结语

简单题