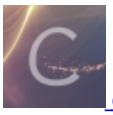
攻防世界Reverse进阶区-re2-cpp-is-awesome-writeup

原创

y4ung 于 2020-10-23 19:25:35 发布 156 收藏 1

分类专栏： ctf 文章标签： ctf



1. 介绍

本题是xctf攻防世界中Reverse的进阶区的题re2-cpp-is-awesome
题目来源： alexctf-2017
题目描述：他们说c++是复杂的，证明他们说的是错的！

2. 分析

```
$ file 5d6da120dfe6499884ac53ef9fe3c80f
5d6da120dfe6499884ac53ef9fe3c80f: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, inter
preter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=08fba98083e7c1f7171fd17c82befdfe1dcbcc82
, stripped
$ chmod +x 5d6da120dfe6499884ac53ef9fe3c80f
$ ./5d6da120dfe6499884ac53ef9fe3c80f
Usage: ./5d6da120dfe6499884ac53ef9fe3c80f flag
$ ./5d6da120dfe6499884ac53ef9fe3c80f 123
Better luck next time
```

扔进IDA里看一下。`Better luck next time\n` 是在sub_400B56函数中被引用的。sub_400B56函数在main函数中被引用。

```
void __noreturn sub_400B56()
{
  std::operator<<<std::char_traits<char>>(&std::cout, "Better luck next time\n");
  exit(0);
}
```

同时注意到 `You should have the flag by now\n` 是在sub_400B73函数中被使用。同样该函数也仅仅是打印了这个字符串。并且sub_400B73函数也是在main中被使用。

在main函数中，首先是将用户输入保存到变量v9中。接下来用一个for循环进行遍历。i初始值为v9的首地址。然后在for循环开始时，验证了循环的退出条件。
当不满足循环退出条件时，往下继续执行。先将i赋值给v6。

```
...

 for ( i = std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::begin(&v9); ; sub_400D7A(&i) )// sub_400D7A  ++i ，注意，这里的i是v9第一个字符的地址
 {
    v11 = std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::end(&v9);// end()函数返回一个迭代器，指向字符串的末尾
    if ( !sub_400D3D(&i, &v11) )                // 循环的退出条件
      break;
    v6 = sub_400D9A(&i);                        // *(&i)=i
    if ( *v6 != off_6020A0[dword_6020C0[v12]] ) // v12每次自增1，在数组0x6020c0（每个数字占4个字节）开始偏移，每次都自增4个字节
                                                // 疑问：IDA里这个0x6020c0开始的字符串是dword，v12自增1，为什么不是一次增加2个字节？
      sub_400B56();                             // Better luck next time\n
    ++v12;
  }
  sub_400B73();
  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(&v9);
  return 0LL;
}
```

最重要的第二个if语句来了。v12是一个索引变量，循环每走一遍，它就自增1，作用是用来获取数组dword_6020C0中的值。然后根据从数组dword_6020C0中得到的值，再作为下标去取字符串0x6020A0的字符，其中字符串0x6020A0为：`L3t_ME_T3ll_Y0u_S0m3th1ng_1mp0rtant_A_{FL4G}_W0nt_b3_3X4ctly_th4t_345y_t0_c4ptur3_H0wev3r_1T_w1ll_b3_C00l_1F_Y0u_g0t_1t` 。而只有当v6，也就是用户输入的当前字符等于从字符串0x6020A0得到的当前字符时，才不会输出`Better luck next time\n` 。

dword_6020C0如下。因此，v12每自增1，相当于dword_6020C0中取4个字节。需要注意的是，文件是小端存储的，也就是对于`0x12345678`这样的数据，内存中是`78 56 34 12`这样存储的。即高位的数据（12）要放在高地址处，低位的数据（78）要放在低地址处。

因此，当v12=0时，取的是`24h 00 00 00`这四个字节，24h是在低地址，在原数据中是在低位，所以数据应该是`0x00000024`

v12=0 => 0 0 0 24h
v12=1 => 0 0 0 0
v12=2 => 0 0 0 5
v12=3 => 0 0 0 35h
… …

```
.data:00000000006020C0 dword_6020C0    dd 24h                  ; DATA XREF: main+DD↑r
.data:00000000006020C4                 align 8
.data:00000000006020C8                 db    5
.data:00000000006020C9                 db    0
.data:00000000006020CA                 db    0
.data:00000000006020CB                 db    0
.data:00000000006020CC                 db  36h ; 6
.data:00000000006020CD                 db    0
.data:00000000006020CE                 db    0
.data:00000000006020CF                 db    0
.data:00000000006020D0                 db  65h ; e
.data:00000000006020D1                 db    0
.data:00000000006020D2                 db    0
.data:00000000006020D3                 db    0
.data:00000000006020D4                 db    7
.data:00000000006020D5                 db    0
.data:00000000006020D6                 db    0
```

```
.data:00000000006020D7                db    0
.data:00000000006020D8                db  27h ; '
.data:00000000006020D9                db    0
.data:00000000006020DA                db    0
.data:00000000006020DB                db    0
.data:00000000006020DC                db  26h ; &
.data:00000000006020DD                db    0
.data:00000000006020DE                db    0
.data:00000000006020DF                db    0
.data:00000000006020E0                db  2Dh ; -
.data:00000000006020E1                db    0
.data:00000000006020E2                db    0
.data:00000000006020E3                db    0
.data:00000000006020E4                db    1
.data:00000000006020E5                db    0
.data:00000000006020E6                db    0
.data:00000000006020E7                db    0
.data:00000000006020E8                db    3
.data:00000000006020E9                db    0
.data:00000000006020EA                db    0
.data:00000000006020EB                db    0
.data:00000000006020EC                db    0
.data:00000000006020ED                db    0
.data:00000000006020EE                db    0
.data:00000000006020EF                db    0
.data:00000000006020F0                db  0Dh
.data:00000000006020F1                db    0
.data:00000000006020F2                db    0
.data:00000000006020F3                db    0
.data:00000000006020F4                db  56h ; V
.data:00000000006020F5                db    0
.data:00000000006020F6                db    0
.data:00000000006020F7                db    0
.data:00000000006020F8                db    1
.data:00000000006020F9                db    0
.data:00000000006020FA                db    0
.data:00000000006020FB                db    0
.data:00000000006020FC                db    3
.data:00000000006020FD                db    0
.data:00000000006020FE                db    0
.data:00000000006020FF                db    0
.data:0000000000602100                db  65h ; e
.data:0000000000602101                db    0
.data:0000000000602102                db    0
.data:0000000000602103                db    0
.data:0000000000602104                db    3
.data:0000000000602105                db    0
.data:0000000000602106                db    0
.data:0000000000602107                db    0
.data:0000000000602108                db  2Dh ; -
.data:0000000000602109                db    0
.data:000000000060210A                db    0
.data:000000000060210B                db    0
.data:000000000060210C                db  16h
.data:000000000060210D                db    0
.data:000000000060210E                db    0
.data:000000000060210F                db    0
.data:0000000000602110                db    2
.data:0000000000602111                db    0
.data:0000000000602112                db    0
```

```
.data:0000000000602112                  db    0
.data:0000000000602113                  db    0
.data:0000000000602114                  db  15h
.data:0000000000602115                  db    0
.data:0000000000602116                  db    0
.data:0000000000602117                  db    0
.data:0000000000602118                  db    3
.data:0000000000602119                  db    0
.data:000000000060211A                  db    0
.data:000000000060211B                  db    0
.data:000000000060211C                  db  65h ; e
.data:000000000060211D                  db    0
.data:000000000060211E                  db    0
.data:000000000060211F                  db    0
.data:0000000000602120                  db    0
.data:0000000000602121                  db    0
.data:0000000000602122                  db    0
.data:0000000000602123                  db    0
.data:0000000000602124                  db  29h ; )
.data:0000000000602125                  db    0
.data:0000000000602126                  db    0
.data:0000000000602127                  db    0
.data:0000000000602128                  db  44h ; D
.data:0000000000602129                  db    0
.data:000000000060212A                  db    0
.data:000000000060212B                  db    0
.data:000000000060212C                  db  44h ; D
.data:000000000060212D                  db    0
.data:000000000060212E                  db    0
.data:000000000060212F                  db    0
.data:0000000000602130                  db    1
.data:0000000000602131                  db    0
.data:0000000000602132                  db    0
.data:0000000000602133                  db    0
.data:0000000000602134                  db  44h ; D
.data:0000000000602135                  db    0
.data:0000000000602136                  db    0
.data:0000000000602137                  db    0
.data:0000000000602138                  db  2Bh ; +
.data:0000000000602139                  db    0
.data:000000000060213A                  db    0
.data:000000000060213B                  db    0
.data:000000000060213B _data            ends
.data:000000000060213B
```

我们可以在IDA中选中dword_6020C0的内容，然后按快捷键：`shift+e`，选择`C unsigned char array (decimal)`。然后将其中的内容复制出来用python脚本提取：

```
text = """
   36,   0,   0,   0,   0,   0,   0,   0,   5,   0,
    0,   0,  54,   0,   0,   0, 101,   0,   0,   0,
    7,   0,   0,   0,  39,   0,   0,   0,  38,   0,
    0,   0,  45,   0,   0,   0,   1,   0,   0,   0,
    3,   0,   0,   0,   0,   0,   0,   0,  13,   0,
    0,   0,  86,   0,   0,   0,   1,   0,   0,   0,
    3,   0,   0,   0, 101,   0,   0,   0,   3,   0,
    0,   0,  45,   0,   0,   0,  22,   0,   0,   0,
    2,   0,   0,   0,  21,   0,   0,   0,   3,   0,
    0,   0, 101,   0,   0,   0,   0,   0,   0,   0,
   41,   0,   0,   0,  68,   0,   0,   0,  68,   0,
    0,   0,   1,   0,   0,   0,  68,   0,   0,   0,
   43,   0,   0,   0
"""  # 124，每个数字都是1个字节
text = [i for i in text.replace(" ", "").replace("\n", "").split(",")]
text_arr = []

for i in range(0, len(text), 4):
    text_arr.append(int(text[i]))
print(text_arr)

target_str = "L3t_ME_T3ll_Y0u_S0m3th1ng_1mp0rtant_A_{FL4G}_W0nt_b3_3X4ctly_th4t_345y_t0_c4ptur3_H0wev3r_1T_w1ll_
b3_C00l_1F_Y0u_g0t_1t"
res = ""
for each in text_arr:
    res += target_str[each]
print(res)  # ALEXCTF{W3_L0v3_C_W1th_CL45535}
```

最终得到flag：`ALEXCTF{W3_L0v3_C_W1th_CL45535}`

3. 总结

对数组的取值，比如本题的`dword_6020C0[v12]`，如有不懂的，可以先动态调试一波，再尝试理解一下。我感觉数组的类型是一个需要关注的点。