Jarvis OJ - DD-Hello -Writeup

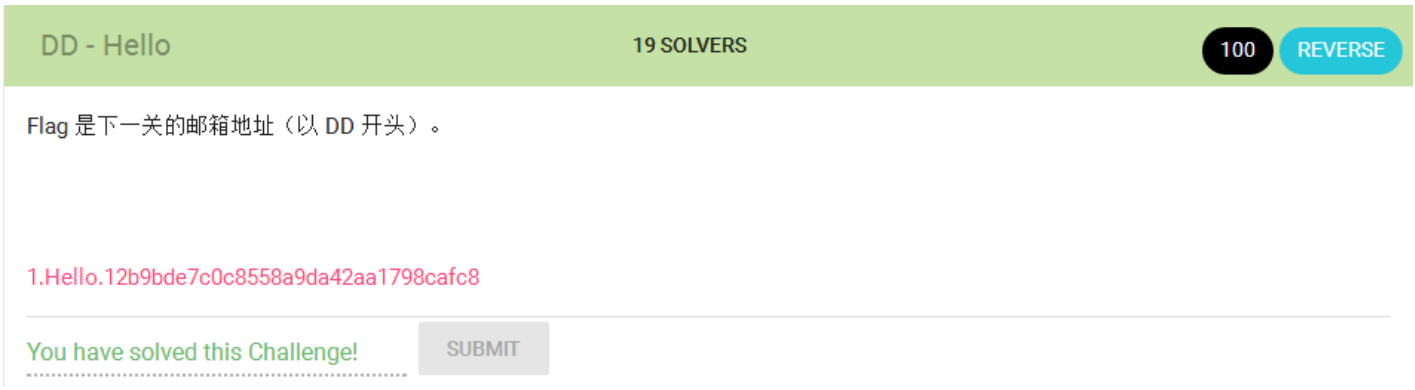
转载

baikeng3674 于 2017-07-26 13:35:00 发布 82 收藏
原文链接：http://www.cnblogs.com/WangAoBo/p/7239216.html
版权

Jarvis OJ - DD-Hello -Writeup

> 转载请注明出处http://www.cnblogs.com/WangAoBo/p/7239216.html

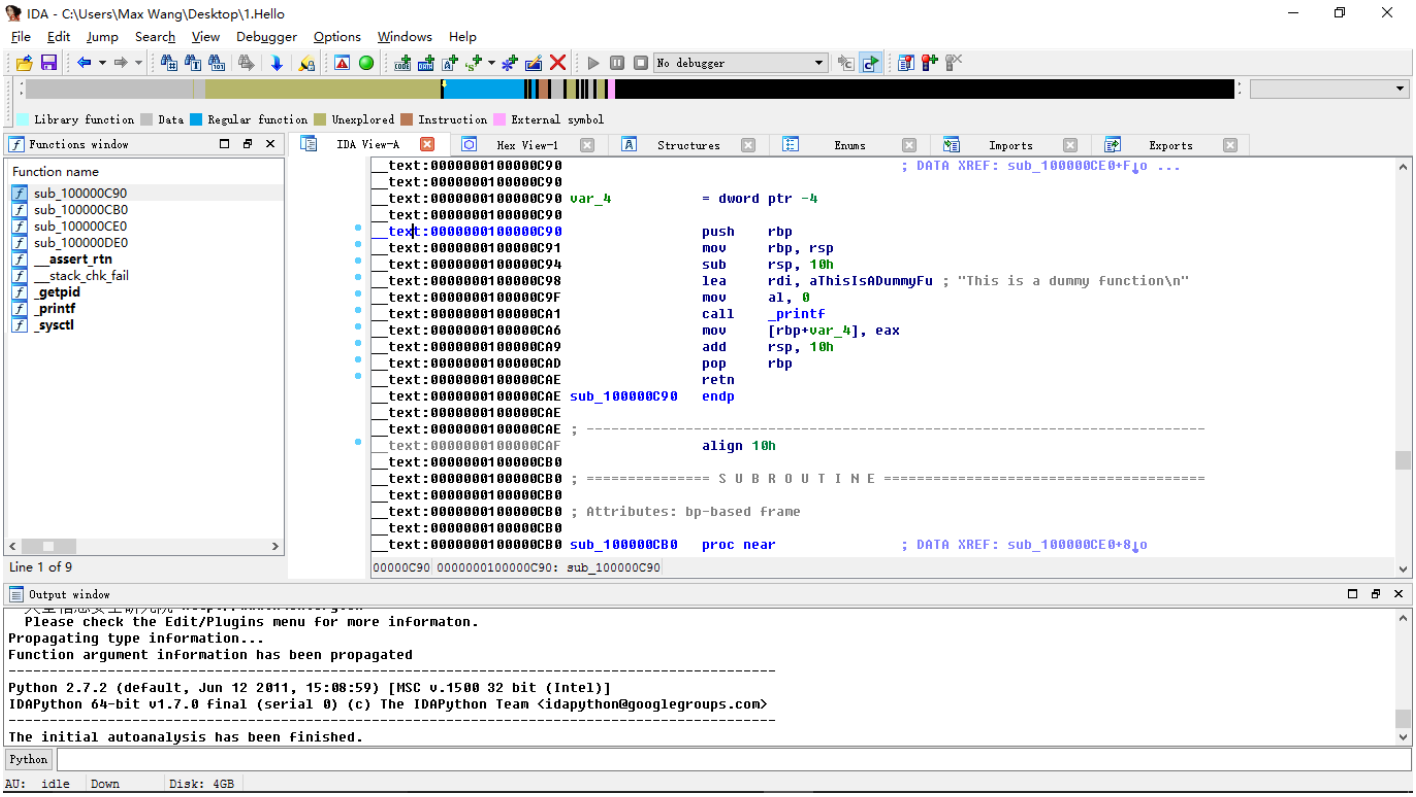
题目：



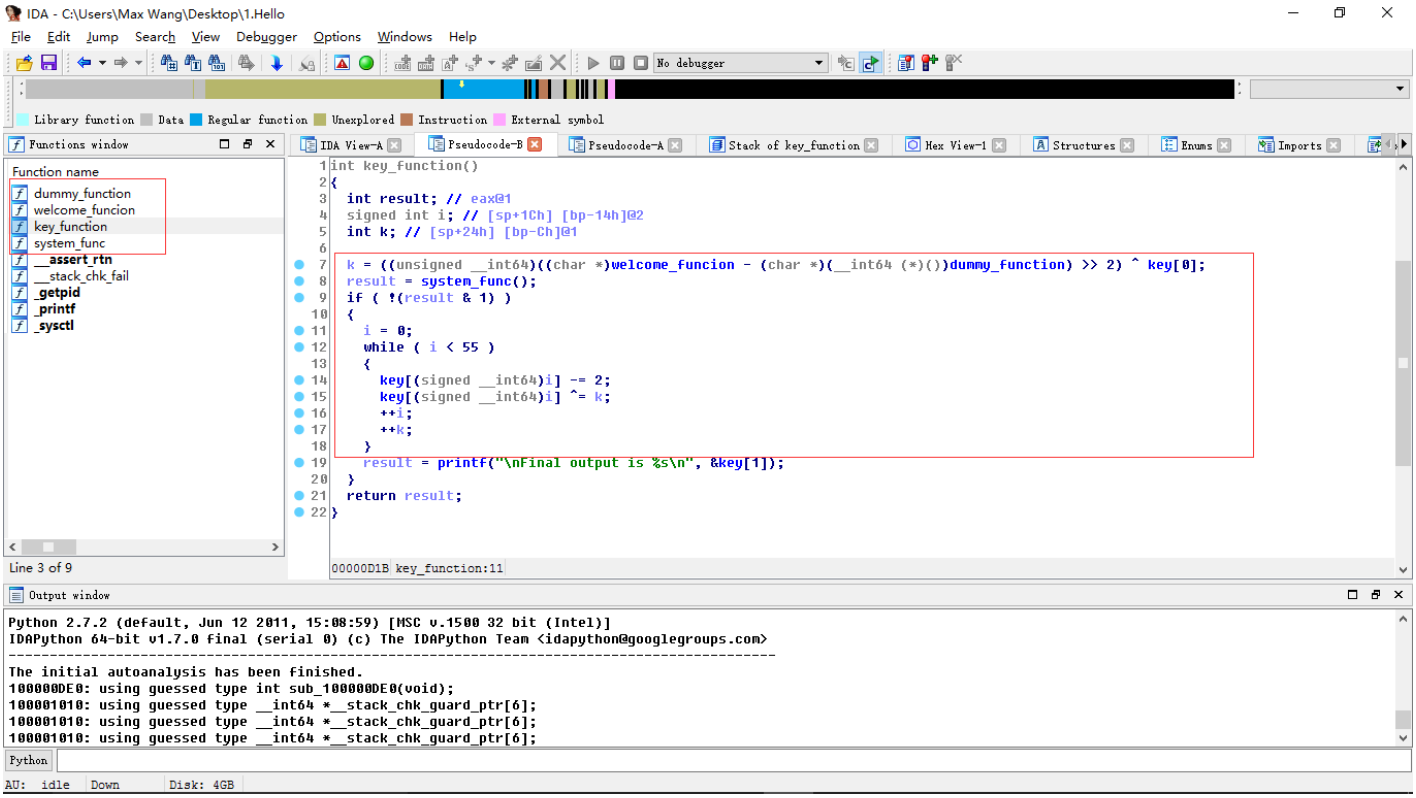
分析：

第一次做这道题时，file查看发现是OS X和IOS的可执行文件，就果断放弃了

后来又一想，OS X是基于FreeBSD的，那Mach-O文件和elf应该也有相通之处，于是就尝试用IDA打开文件，居然真的打开了



函数较少，并且都可F5出伪代码，逐个分析，整理如下：



关键代码：

```
if ( !(result & 1) )
{
  i = 0;
  while ( i < 55 )
  {
    key[(signed __int64)i] -= 2;
    key[(signed __int64)i] ^= k;
    ++i;
    ++k;
  }
```

key[]是已知的，i是循环变量，因此只需要找到k的值，从伪代码看不出什么

```
k = ((unsigned __int64)((char *)welcome_funcion - (char *)(__int64 (*)())dummy_function) >> 2) ^ key[0];
```

但k肯定是固定的，因此首先尝试爆破，脚本如下：

```
for kk in range(0, 256):
    key = [0x41, 0x10, 0x11, 0x11, 0x1B, 0x0A, 0x64, 0x67, 0x6A, 0x68, 0x62, 0x68, 0x6E, 0x67, 0x68, 0x6B,
0x62, 0x3D, 0x65, 0x6A, 0x6A, 0x3D, 0x68, 0x4, 0x5, 0x8, 0x3, 0x2, 0x2, 0x55, 0x8, 0x5D, 0x61, 0x55, 0x0A,
0x5F, 0x0D, 0x5D, 0x61, 0x32, 0x17, 0x1D, 0x19, 0x1F, 0x18, 0x20, 0x4, 0x2, 0x12, 0x16, 0x1E, 0x54, 0x20,
0x13, 0x14, 0x0, 0x0]
    try:
        k = kk
        i = 0
        while i < 55:
            key[i] -= 2
            key[i] ^= k

            i += 1
            k += 1

        ans = ''
        for c in key:
            ans += chr(c)

        print ans

    except:
        pass
```

运行并提取flag如下：

```
[max@parrot]-[~/Desktop]
 $python hello.py | strings | grep DDCTF
DDCTF-5943293119a845e9bbdbde5a369c1f50@didichuxing.com
```

后来请教了学长，从汇编代码里找出了k的值，就不用爆破了

```
8 ; 6:   k = ((unsigned __int64)((char *)welcome_funcion - (char *)(__int64 (*)())dummy_function) >> 2) ^ key[0];
8                lea     rax, welcome_funcion ; 前三行是无用代码
F                lea     rcx, dummy_function ; 前三行是无用代码
6                sub     rax, rcx        ; 前三行是无用代码
9                mov     rcx, 4          ; 第4行的操作与k值无关
3                movsx   edx, cs:key     ; key[0]赋给edx寄存器
A                mov     [rbp+var_4], edx ; rbp+var_4 == edx == key[0]
D                lea     rsi, dummy_function ; 取dummy_function的地址存到rsi寄存器
4                lea     rdi, welcome_funcion ; 取welcome_function的地址存到rdi寄存器
B                sub     rdi, rsi        ; rdi = rdi - rsi
E                shr     rdi, 2          ; rdi逻辑右移2位，对应>>2，此时edi为rdi的低32位，即edi == rdi
2                mov     edx, edi        ; edx = edi
4                mov     [rbp+var_8], edx ; rbp+var_8 = edx = edi
7                mov     edx, [rbp+var_4] ; edx = rbp+var_4 = key[0]
A                xor     edx, [rbp+var_8] ; edx = edx ^ rbp+var_8
D                mov     [rbp+k], edx    ; rbp+k = edx,此句为给k赋值的汇编指令
0 ; 7:   result = system_func();
```

于是k即为*dummy_function*与*welcome_function*的地址差值右移两位在与key[0]亦或，从IDA中可以看出*dummy_function*和*welcome_function*的地址

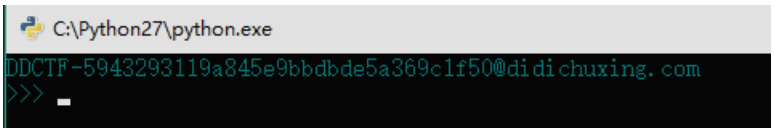
```
00100000C90
00100000C90 dummy_function  proc near               ; CODE XREF: welcome_funcion+20↓p
00100000C90                                         ; DATA XREF: key_function+F↓o ...
00100000C90
00100000C90 var_4           = dword ptr -4
00100000C90
00100000C90                 push    rbp
00100000C91                 mov     rbp, rsp
00100000C94                 sub     rsp, 10h
00100000C98 ; 2:   return printf("This is a dummy function\n");
00100000C98                 lea     rdi, aThisIsADummyFu ; "This is a dummy function\n"
00100000C9F                 mov     al, 0
00100000CA1                 call    _printf
00100000CA6                 mov     [rbp+var_4], eax
00100000CA9                 add     rsp, 10h
00100000CAD                 pop     rbp
00100000CAE                 retn
00100000CAE dummy_function  endp
00100000CAE
```

```
__text:0000000100000CB0 welcome_funcion proc near               ; DATA XREF: key_function+8↓o
__text:0000000100000CB0                                         ; key_function+34↓o
__text:0000000100000CB0
__text:0000000100000CB0 var_8           = dword ptr -8
__text:0000000100000CB0 var_4           = dword ptr -4
__text:0000000100000CB0
__text:0000000100000CB0                 push    rbp
__text:0000000100000CB1                 mov     rbp, rsp
__text:0000000100000CB4                 sub     rsp, 10h
__text:0000000100000CB8 ; 2:   printf("Welcome\n");
__text:0000000100000CB8                 lea     rdi, aWelcome   ; "Welcome\n"
__text:0000000100000CBF                 mov     [rbp+var_4], 0
__text:0000000100000CC6                 mov     al, 0
__text:0000000100000CC8                 call    _printf
__text:0000000100000CCD ; 3:   dummy_function();
__text:0000000100000CCD                 mov     [rbp+var_8], eax
__text:0000000100000CD0                 call    dummy_function
__text:0000000100000CD5 ; 4:   return 0LL;
__text:0000000100000CD5                 xor     eax, eax
__text:0000000100000CD7                 add     rsp, 10h
__text:0000000100000CDB                 pop     rbp
__text:0000000100000CDC                 retn
__text:0000000100000CDC welcome_funcion endp
```

得到新的非爆破脚本如下：

```
key = [0x41, 0x10, 0x11, 0x11, 0x1B, 0x0A, 0x64, 0x67, 0x6A, 0x68, 0x62, 0x68, 0x6E, 0x67, 0x68, 0x6B, 0x62,
0x3D, 0x65, 0x6A, 0x6A, 0x3D, 0x68, 0x4, 0x5, 0x8, 0x3, 0x2, 0x2, 0x55, 0x8, 0x5D, 0x61, 0x55, 0x0A, 0x5F,
0x0D, 0x5D, 0x61, 0x32, 0x17, 0x1D, 0x19, 0x1F, 0x18, 0x20, 0x4, 0x2, 0x12, 0x16, 0x1E, 0x54, 0x20, 0x13,
0x14, 0x0, 0x0]

k = ((0x0000000100000CB0 - 0x0000000100000C90) >> 2) ^ key[0]
i = 0
while i < 55:
    key[i] -= 2
    key[i] ^= k

    i += 1
    k += 1

ans = ''
for c in key:
    ans += chr(c)

print ans[1: ]
```

运行



于是flag即为**DDCTF-5943293119a845e9bbdbde5a369c1f50@didichuxing.com**

转载于:https://www.cnblogs.com/WangAoBo/p/7239216.html