XNUCA-RE WRITEUP

YHZX_2013

bluecake

A1kaid

2016/8/27

Indirect

算法集中在，很容易写出逆算法：

```
0108127E  .  0FBE0C01     movsx ecx,byte ptr ds:[ecx+eax]      success[i]
01081282  .  33D1         xor edx,ecx                          (flag[i]>>4)^success[i]
01081284  .  66:8955 F8   mov word ptr ss:[ebp-0x8],dx
01081288  .  0FBF55 FC    movsx edx,word ptr ss:[ebp-0x4]
0108128C  .  0355 F0      add edx,[local.4]                    (flag[i]%0x10) ^ sorry[i] + i
0108128F  .  B8 04000000  mov eax,0x4
01081294  .  C1E0 00      shl eax,0x0                          0x4
01081297  .  B9 04000000  mov ecx,0x4
0108129C  .  6BC9 00      imul ecx,ecx,0x0                     0x0
0108129F  .  8B75 08      mov esi,[arg.1]
010812A2  .  8B0406       mov eax,dword ptr ds:[esi+eax]
010812A5  .  8B0C08       mov ecx,dword ptr ds:[eax+ecx]
010812A8  .  8B45 F0      mov eax,[local.4]                    i
010812AB  .  3B1481       cmp edx,dword ptr ds:[ecx+eax*4]
010812AE  v  75 2C        jnz short indirect.010812DC
010812B0  .  0FBF4D F8    movsx ecx,word ptr ss:[ebp-0x8]
010812B4  .  2B4D F0      sub ecx,[local.4]                    (flag[i]>>4)^success[i] - i
010812B7  .  BA 04000000  mov edx,0x4
```

但是……主办方给的程序 check 加和减是反的，导致只有 i=0 的时候才能通过 check，到第五位根本无解。被坑了 1 个小时后，人工脑补出正确 check 算法得到 flag（OTZ）：

```
def p():
    hig1 = [0x43, 0x6f, 0x6e, 0x67, 0x72, 0x61, 0x74, 0x75, 0x6c, 0x61, 0x74, 0x69, 0x6f, 0x6e, 0x21]
    cmp1 = [0x40, 0x6a, 0x6b, 0x63, 0x75, 0x6c, 0x77, 0x79, 0x77, 0x6f, 0x81, 0x79, 0x74, 0x7a, 0x30]
    low1 = [0x73, 0x6f, 0x72, 0x72, 0x79, 0x2c, 0x70, 0x6c, 0x65, 0x61, 0x73, 0x65, 0x20, 0x74, 0x72]
    cmp2 = [0x72, 0x65, 0x74, 0x73, 0x74, 0x24, 0x79, 0x67, 0x5e, 0x5e, 0x66, 0x5c, 0x17, 0x6a, 0x65]

    s = ''
    for i in range(0, 15):
        high = ((hig1[i] ^ (cmp1[i]-i))) % 16
        low = ((low1[i] ^ (cmp2[i]+i))) % 16
        #print high, low
        s += chr(high * 16 + low)

    print s
```

```
Python 2.7.11 Shell
File  Edit  Shell  Debug  Options  Window
Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec
AMD64)] on win32
Type "copyright", "credits" or "license(
>>>
========================= RESTART: D:\CT
1itt1e_r3v3rs31
```

运行结果：

```
D:\CTF\X>indirect.exe
Please input flag:1itt1e_r3v3rs31
Congratulation! you got a right flag!
sorry,please try again!
```

indirect.exe

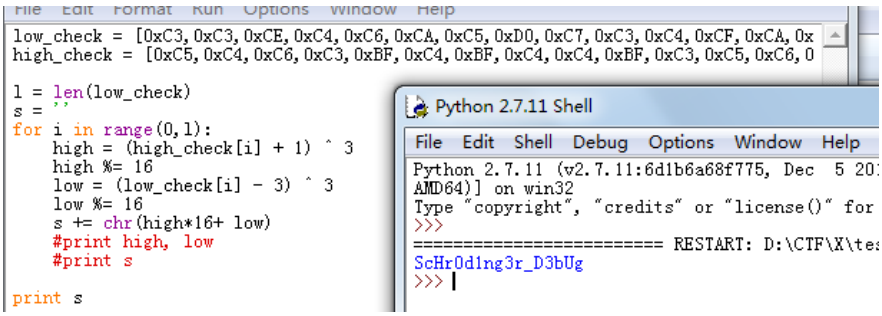
indirect.exe 已停止工作

Schrodingers_Debug

这题有一堆 debug 检测和一个假的 check 函数（解出来好像是 this is no flag 之类）。反调试检测太多，直接断在函数入口处，然后修改 EIP 到真正的 check 函数 sub_401890：

```
for ( i = 0; i < 17; ++i )
{
  if ( (signed __int16)(**(_WORD **)(*(&dword_40413C + 1) + 12) ^ v2[i] & 0xF) + 3 != dword_404090[i]
    || (signed __int16)(**(_WORD **)(*(&dword_40413C + 1) + 12) ^ (v2[i] >> 4)) - 1 != dword_4040D8[i] )
  {
    sub_401CA0(*(_DWORD *)(*(&dword_40413C + 2) + 8));
    exit(0);
  }
}
```

算法基本上和 indirect 一致，写出解密脚本得到 flag：



ezpz

丢 IDA，看到的函数主体

```
puts("Welcome to the world of reversing!");
puts("input the flag:");
setvbuf(stdin, 0, 2, 0);
setvbuf(stdout, 0, 2, 0);
__isoc99_scanf("%s", &s);
if ( strlen(&s) != 32 || check((int)&s) )
{
  puts("try again!");
  result = 0;
}
else
{
  puts("Good job!");
  result = 0;
}
```

```
int __cdecl check(int a1)
{
  int result; // eax@1
  int v2; // edx@1
  char s1; // [sp+1Ch] [bp-5Ch]@1
  int v4; // [sp+6Ch] [bp-Ch]@1

  v4 = *MK_FP(__GS__, 20);
  Base64encode(&s1, a1, 32);
  result = strcmp(&s1, res);
  v2 = *MK_FP(__GS__, 20) ^ v4;
  return result;
}
```

看到是 Base64 加密了字符串，结果是 res，

```
.data:0804B060 ; char res[]
.data:0804B060 res             db 'c3ViOXRxZHl4dncycXdqYjVod2RvMXAzbXp0NDF3cTQ=',0
```

找到了 res

解密得 flag

Transformation

扔 IDA，看到这样的字符串

```
.data:00410AE8 flag_r:          .word aS1mp1e_mips_0n    # "S1mp1e_mips_0ne_by_one"
```

感觉可能是 flag，结果还真是。。

Basyandroid

这个题必须吐槽，官方工作效率太低，搞了一个早上才把错误的题目换掉！！！

言归正传，先来看一下代码

```java
public String check(String arg8) {
    byte[] v2 = arg8.getBytes();
    int v1;
    for(v1 = 0; v1 < v2.length; ++v1) {
        v2[v1] = ((byte)(v2[v1] ^ v1));  // 与下标进行异或
    }

    byte[] v0 = new byte[]{5, 9, 14, 1, 84, 15, 15, 84, 8, 9, 6, 88, 1, 15, 89, 7, 27, 29, 68, 67};
    for(v1 = 0; v1 < v2.length; ++v1) {
        v2[v1] = ((byte)(v2[v1] ^ v0[v1]));  // 再来次一次异或
    }

    byte[] v3 = new byte[30];
    for(v1 = 0; v1 < v2.length; v1 += 2) {
        v3[v1] = ((byte)((v2[v1] - 48) * 10 + (v2[v1 + 1] - 48)));  // 将两位压缩成一位
    }

    for(v1 = 0; v1 < v3.length; ++v1) {
        v3[v1] = ((byte)(v3[v1] + (((byte)(v3.length - v1))) * 3));
    }

    StringBuilder v4 = new StringBuilder();
    for(v1 = 0; v1 < v3.length; ++v1) {
        v4.append(v3[v1]);
    }

    return v4.toString();
```

最终生成的结果再来和 x-nuca2016 进行对比，搜索空间不大，所以直接正向爆破

```
import string

username = "x-nuca2016"

password = range(20)

#v1 = [5, 9, 14, 1, 84, 15, 15, 84, 8, 9, 6, 88, 1, 15, 89, 7, 27, 29, 68, 67]

v1 = [13, 9, 6, 9, 93, 1, 14, 84, 9, 14, 6, 91, 10, 5, 89, 6, 28, 23, 69, 65]

space = "0123456789abcdef"

solution = [[] for i in range(10)]

for i in range(10):

    for c1 in space:

        for c2 in space:
```

```
            c = ((ord(c1) ^ (2*i) ^ v1[2*i]) - 48) * 10
            c += (ord(c2) ^ (2*i+1) ^ v1[2*i+1]) - 48


            c = c + (10 - i)*3
            if c < 256 and c > 0:
                tc = chr(c)
            else:
                tc = "?"
            if tc == username[i]:
                solution[i].append(c1 + c2)
print solution
flags = solution[0]
for s in solution[1:10]:
    tmp_flag = []
    for f in flags:
        for ss in s:
            tmp_flag.append(f + ss)
    flags = tmp_flag
```

for i,f in enumerate(flags):

print i,f

lookatme

同样是一个算法题，仔细分析就可以了

Java 部分

```
public void onClick(View arg5) {
    if(MainActivity.check(MainActivity.this.findViewById(2131165185).getText().toString()) == 1) {
        Toast.makeText(MainActivity.this.getApplicationContext(), "you got it", 1).show();
    }
    else {
        Toast.makeText(MainActivity.this.getApplicationContext(), "try try try again ~", 1).show();
    }
}
```

在 ida 中打开 libCrackme，使用 check 关键字搜索函数

signed int __fastcall Java_oct_rd6_iie_crackme_MainActivity_check(JNIEnv_ *env, int a2, int jstring_flag)

{

_env = env;

v4 = env->functions;

v24 = _stack_chk_guard;

v5 = (const char *)((int (__fastcall *)(JNIEnv_ *, int, _DWORD))v4->GetStringUTFChars)(_env, jstring_flag, 0);

length = j_j_strlen(v5);

if (length > 14) // 长度大于 14

```
{
  flag_1 = j_operator new[](20u);
  j_j_strcpy((char *)flag_1, v5);
  swap(flag_1, (char *)(flag_1 + 8));
  swap(flag_1 + 1, (char *)(flag_1 + 3));
  swap(flag_1 + 7, (char *)(flag_1 + 2));
  swap(flag_1 + 11, (char *)(flag_1 + 3));
  swap(flag_1 + 14, (char *)(flag_1 + 9));
  swap(flag_1 + 4, (char *)(flag_1 + 10));      // 置换
  j_j_memcpy(&dest, &unk_23C4, 0x2Bu);
  v8 = j_j_strlen(&dest);
  dec(&dest, v8, 0x80u);
  fp = j_j_fopen(&dest, (const char *)&unk_2404);// /data/data/oct.rd6.iie.crackme/files/cache
  if ( fp )
  {
    v9 = 0;
    byte_1 = 0;
    j_j_fread(&byte_1, 3u, 1u, fp);
```

```
j_j_fseek(fp, byte_1 ^ 0xFFDC30, 1);          // offset = 0x4cf

v19 = 0;

j_j_fread(&v19, 1u, 1u, fp);

do

  *(_BYTE *)(flag_1 + v9++) ^= v19;

while ( v9 != length );                        //  所有元素与 cache 的第 x 个元素异或

if ( check1((char *)flag_1) )

{

  jCLass = ((int (__fastcall *)(JNIEnv_ *, const char *))_env->functions->FindClass)(

                    _env,

                    "oct/rd6/iie/crackme/CrackmeApp");

  jMethod = ((int (__fastcall *)(JNIEnv_ *, int, const char *, const char
*))_env->functions->GetStaticMethodID)(

                     _env,

                     jCLass,

                     "onHighMemory",

                     "(JIB)I");

  j_j_memcpy(v21, &unk_23EF, 4u);              // 0x3, 0x57, 0x50, 0xe
```

```
v22[0] = 0x16156;

v22[1] = 0x1AC29;

v22[2] = 0x185A9;

j = 0;

v22[3] = 0x3C48D;

flag_ptr = flag_1;

while ( 1 )

{

  arg1 = (*(_BYTE *)flag_ptr << 24)        //  可以确定

         + (*(_BYTE *)(flag_ptr + 1) << 16)

         + (*(_BYTE *)(flag_ptr + 2) << 8)

         + *(_BYTE *)(flag_ptr + 3);          //  四个字节拼凑成一个 dword

  arg2 = v22[j];

  arg3 = (unsigned __int8)v21[j];

  if ( !_JNIEnv::CallStaticIntMethod(_env, jCLass, jMethod) )

    break;

  ++j;

  flag_ptr += 4;
```

```
                if ( j == 4 )                                   //  总共进行四轮，所以 flag 总共 16 个字节
                {
                  j_j_fread(&byte_1, 1u, 1u, fp);
                  if ( !sub_8088((char *)flag_1, byte_1) )
                    break;
                  j_j_fclose(fp);
                  result = 1;
                  goto success;
                }
              }
            }
          }
        }
    result = 0;
success:
    if ( v24 != _stack_chk_guard )
      j_j___stack_chk_fail(result);
    return result;
```

}

代码中通过反射调用了 CrackMeAPP 中的 onHighMemory 函数

```
public static int onHighMemory(long arg10, int arg12, byte arg13) {
    int v5 = 0;
    byte v0 = ((byte)(((int)(arg10 >> 24 & 255))));
    byte v1 = ((byte)(((int)(arg10 >> 16 & 255))));
    byte v2 = ((byte)(((int)(arg10 >> 8 & 255))));
    byte v3 = ((byte)(((int)(arg10 >> 0 & 255))));
    int v4 = v2 * 12 * v2 + v3 * 13 * v3 + v2 * 14 * v3 + v1 * 15 * v0 + 16;
    if((v0 ^ arg13) == v1 && v4 == arg12) {
        v5 = 1;
    }

    return v5;
}
```

同样，本题比较坑的是在进行交换的时候两次涉及到同一个位置

```
swap(flag_1, (char *)(flag_1 + 8));
swap(flag_1 + 1, (char *)(flag_1 + 3));
swap(flag_1 + 7, (char *)(flag_1 + 2));
swap(flag_1 + 11, (char *)(flag_1 + 3));
swap(flag_1 + 14, (char *)(flag_1 + 9));
swap(flag_1 + 4, (char *)(flag_1 + 10));
j j memcpy(&dest, &unk 23C4, 0x2Bu);
```

一开始没有注意到浪费了很多去检查代码，好吧，看看代码

import string

arg2 = [0x16156, 0x1AC29, 0x185A9, 0x3C48D]

arg3 = [0x3, 0x57, 0x50, 0xe]

flag1 = [0x0 for i in range(16)]

flag1[0] = 70

```
flag1[4] = 68

flag1[8] = 18

# flag1[12] <= 65




sums = []      # 12*(v2^2) + 13*(v3^2) + 14*v3*v2

#reverse onHighMemory, 4 bits as a unit

for i in range(3):

        v0 = flag1[4*i]

        v1 = v0 ^ arg3[i]

        sums.append(arg2[i] - 15*v0*v1 - 16)

        flag1[4*i+1] = v1



print flag1

print sums



#reverse sub_8088

'''
```

```
        byte1 = (flag1[3] + byte1) % 128

        byte2 = (flag1[7] + byte1) % 128

        byte3 = (flag1[11] + byte1) % 128

        byte4 = (flag1[15] + byte1) % 128

        finally

                byte1 = 58

                byte2 = 109

                byte3 = 105

                byte[4] <= 104

'''

bytes_new = [58, 109, 105, 104]

byte1 = 0xa6



for i in range(3):

        for v2 in range(256):

                for v3 in range(256):

                        if 12*v2*v2 + 13*v3*v3 + 14*v3*v2 == sums[i]:

                                byte1_new = (v3 + byte1) % 128
```

```
                #print v2, v3,byte1_new
                if byte1_new == bytes_new[i]:
                    flag1[4*i+2] = v2
                    flag1[4*i+3] = v3
print flag1


flag1[12] = []
flag1[13] = []
flag1[14] = []
flag1[15] = []
for v0 in range(66):
    v1 = v0 ^ arg3[3]
    tsum = arg2[3] - 15*v0*v1 - 16
    for v2 in range(256):
        for v3 in range(256):
            if 12*v2*v2 + 13*v3*v3 + 14*v3*v2 == tsum:
                byte1_new = (v3 + byte1) % 128
                if byte1_new <= bytes_new[3]:
```

```
                flag1[12].append(v0)

                flag1[13].append(v1)

                flag1[14].append(v2)

                flag1[15].append(v3)



print flag1



def isChars(flag):

    for i in flag:

        if i not in '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ_':

            return False

    return True



byte_xor = 0x76

flag2 = []
```

```
for i,f in enumerate(flag1):
    if i<12:
        flag2.append(flag1[i] ^ byte_xor)
    else:
        tmpL = []
        for t in flag1[i]:
            tmpL.append(t ^ byte_xor)
        flag2.append(tmpL)
#print flag2


def swap(i,j):
    tmp = flag2[i]
    flag2[i] = flag2[j]
    flag2[j] = tmp


swap(0,8)
swap(11,3)
swap(1,3)
```

```
swap(7,2)

swap(14,9)

swap(4,10)



# print flag2



for i in range(len(flag1[12])):

    flag = ""

    for k in flag2:

        if type(k) == list:

            flag += chr(k[i])

        else:

            flag += chr(k)

    if isChars(flag):

        print byte_xor

        print flag.encode('hex'),flag
```