# VolgaCTF2015之lcg的writeup

原创

HappyOrange2014 于 2015-05-04 11:07:34 发布 1292 收藏

分类专栏： CTF 文章标签： CTF crypto



本文链接：https://blog.csdn.net/HappyOrange2014/article/details/45478199





题目描述：

Check out our brand-new cipher. It’s super-fast and moderately secure.

```python
#!/usr/bin/python
import struct
import os

M = 65521
class LCG():
    def __init__(self, s):
        self.m = M
        (self.a, self.b, self.state) = struct.unpack('<3H', s[:6])

    def round(self):
        self.state = (self.a*self.state + self.b) % self.m
        return self.state

    def generate_gamma(self, length):
        n = (length + 1) / 2
        gamma = ''
        for i in xrange(n):
            gamma += struct.pack('<H', self.round())
        return gamma[:length]

def encrypt(data, key):
    assert(len(key) >= 6)
    lcg = LCG(key[:6])
    gamma = lcg.generate_gamma(len(data))
    return ''.join([chr(d ^ g)  for d,g in zip(map(ord, data), map(ord, gamma))])

def decrypt(ciphertext, key):
    return encrypt(ciphertext, key)

def sanity_check():
    data = 'A'*100
    key = os.urandom(6)
    ciphertext = encrypt(data, key)
    decrypted_data = decrypt(ciphertext, key)
    assert(data == decrypted_data)

if __name__ == '__main__':
    with open('flag.png', 'rb') as f:
        data = f.read()
    key = os.urandom(6)
    enc_data = encrypt(data, key)
    with open('flag.enc.bin', 'wb+') as f:
        f.write(enc_data)
```

本题给出了一份python加密程序的源码和一个密文文件，需要通过源码逆向出密钥key来破解以获得flag。
解题的关键点如下：

1、通过加密文件名flag.png.bin可知原文件为png图片，从而可知明文的前8个字节为**0x89504E470D0A1A0A**，即png图片的固定文件头。而密文的前8个字节为**0x5FC79CA8BCE94A78**。这两组8字节是逆向出密钥key的基础。

2、加密过程实际是一个XOR过程，在破解出密钥key后，只需利用原加密程序重新XOR一次即可，无需自行编写程序。

3、理解python的struct库的pack和unpack函数。Struct库是python用来解析数据的工具：

**struct.pack('H', x)**表示（H前有个<符号，不知为啥就是显示不出来？），将x按照H格式，以小端优先的规则解析。其中，H表示无符号短整型（两字节），网上有完整的类型表，可查阅；而“<”表示little-endian，即小端优先。举例如下：

```
gamma = struct.pack('<H', 38870)#38870的16进制为97d6

print "gamma : ", binascii.b2a_hex(gamma)#结果为d697
```

**struct.unpack('<3H', x)**表示，将x以小端优先的规则还原成H格式，结果以tuple形式输出。其中，3表示需还原3个无符号短整型（相当于6字节）。举例如下：

```
p = struct.unpack('<3H', "0d1d0a") #共分成三个元素，即0d、1d、0a，每个元素按照小端优先（即d0、d1、a0）的原则还原

print p #结果为(25648,25649,24880)，相当于16进制的（6430，6431，6130），即（d0，  d1，a0）
```

4、加密程序本质上就是，将明文数据与通过密钥key生成的gamma字符串按字节XOR得到密文，而gamma字符串的计算过程可简写为：

```
self.state = (self.a*self.state + self.b) % self.m  (*)
n = (datalength + 1) / 2
gamma = ''
for i in xrange(n):
    gamma += struct.pack('<H', self.state)
```

其中，m是常数65521。a、b和state是6字节密钥key通过struct.unpack('<3H', x)获得的三元tuple，而state会根据（*）式迭代更新，每一轮的state会带入下一轮的计算。

同时，尽管n仅为明文数据长度的一半，但由于struct.pack解析出的数据均为H格式（双字节），故gamma的长度与明文长度是一致的。

5、根据1可知，gamma字符串的前8字节为明文与密文的XOR结果：
明文：0x89504E470D0A1A0A
密文：0x99CE83E95DE0D8E0
Gamma：0x109ecdae50eac2ea
故可得下面等式：
65521 * x + 40464 = a * s + b （40464即109e的小端优先9e10的对应整数）
65521 * y + 44749 = a * 40464 + b （上一轮的余数40464作为s带入下一轮）
65521 * z + 59984 = a * 44749 + b
65521 * w + 60098 = a * 59984 + b
从而可得：
65521 * (w-z) + 114 = a * 15235
又知a的取值范围为（0，65535），可通过以下代码爆破出a，计算可得唯一解a=44882。
同理可得b=50579，s=37388。

```
for a in range(0, 65535):
    tmp = a * 15235 - 114
    if tmp % 65521 == 0:
        print "a : ", a
```

6、由a、b、s，根据struct的规则可得：
a=44882->AF52->52AF
b=50579->C593->93C5
s=37388->920C->0C92
密钥key为：52AF93C50C92
带入原加密程序即可还原图片得到flag。

{linear_congruential_generator_isn't_good_for_crypto}

Flag：{linear_congruential_generator_isn’t_good_for_crypto}