Crypto之击破多层加密

一叶飘零 / 2018-08-24 14:46:08 / 浏览数 2393

前言

最近在看Crypto，记录一下一道题的过程
题目链接

```
https://hackme.inndy.tw/static/multilayer.7z
```

注：似乎平台不支持数学公式，想看公式的可以参见
http://skysec.top/2018/08/24/Crypto%E4%B9%8B%E5%87%BB%E7%A0%B4%E5%A4%9A%E5%B1%82%E5%8A%A0%E5%AF%86/#%E5%90%8E%E8%AE%B0

题干分析

题目名称叫多层，所以我们看到最后的加密形式为

```
flag = layer1(flag)
flag = layer2(flag)
flag = layer3(flag)
flag = layer4(flag)

print(flag.decode('ascii'))
```

那么我们逐层分析

layer1

```
def layer1(data):
   data = data.decode('ascii')
   s = string.ascii_uppercase
   t = list(s)
   random.shuffle(t)
   t = ''.join(t)
   print(collections.Counter(data))
   return data.translate(str.maketrans(s, t)).encode('ascii')
```

s的输出为

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

t将s随机打乱重新排列
然后将题目将flag中出现的字频统计出来，打印出来，又将flag中的字母按s-t这个替换表替换
例如：

```
s■ABCDEFGHIJKLMNOPQRSTUVWXYZ
t■ NVOJWAXCBQRSMZYUDKILEFTPHG
```

按这个对应顺序替换（当然t是随机打乱的，这只是个例子）,最后再全部转换为ascii编码
我们看一下flag的字频统计：

```
Counter({' ': 14, 'O': 6, 'A': 5, 'U': 4, 'I': 4, 'T': 4, 'N': 3, 'D': 3, 'E': 3, 'L': 3, 'H': 3, 'Y': 3, 'R': 3, 'G': 2, 'C':
```

layer2

第二层很短

```
def layer2(data):
   return bytes(b * 17 % 251 for b in data)
```

将layer1的结果乘17取余251，这也很好反求回来
$$
17b \equiv c \text{ } \mod \text{ } 251\
$$
问题即：我们已知c怎么求b
这里用逆元即可：
$$

b \equiv c17^{-1} \text{ } mod \text{ } 251\
$$
17的逆元也很好求，如下：

```
import gmpy2
print gmpy2.invert(17,251)
```

即可得到192
故此，这一层我们可以轻松取逆
$$
b \equiv c*192 \text{ } mod \text{ } 251\
$$

layer3

```
def layer3(data):
   output = []
   key = number.bytes_to_long(os.urandom(128))

   for i in data:
       key = (key * 0xc8763 + 9487) % 0x10000000000000000
       output.append((i ^ key) & 0xff)
   return bytes(output)
```

这一步也很简单，将layer2的结果用同一个key进行异或加密，其中key是最开始随机生成的
乍一看，这里的key无法预测，极大，爆破无解，但是仔细分析，发现只是纸老虎
我们观察到对key的处理

```
key = (key * 0xc8763 + 9487) % 0x10000000000000000
output.append((i ^ key) & 0xff)
```

这里的

```
(i ^ key) & 0xff
```

是可以使用分配律的，可以拆分为

```
(i&0xff)^(key&0xff)
```

这样一来，我们的key就变成了0~256的一个极小的数，简单爆破即可
并且因为i一定在0x00~0xff这个范围内，所以这一层的求逆也变得很容易

```
layer3(c,x)
```

即可，其中
$$
x \in （0，256）
$$

layer4

来到最后一层

```
def layer4(data):
   iv = os.urandom(256)
   output = iv

   hexencoded = binascii.hexlify(data)
   length_target = (len(hexencoded) + 3) // 4
   padded = hexencoded.ljust(length_target * 4, b'f')

   for i in range(0, len(padded), 4):
       r = rsa_encrypt(padded[i:i+4])
       block = binascii.unhexlify('%.512x' % r)
       output += xor(output[-256:], block)

   return base64.b64encode(output)
```

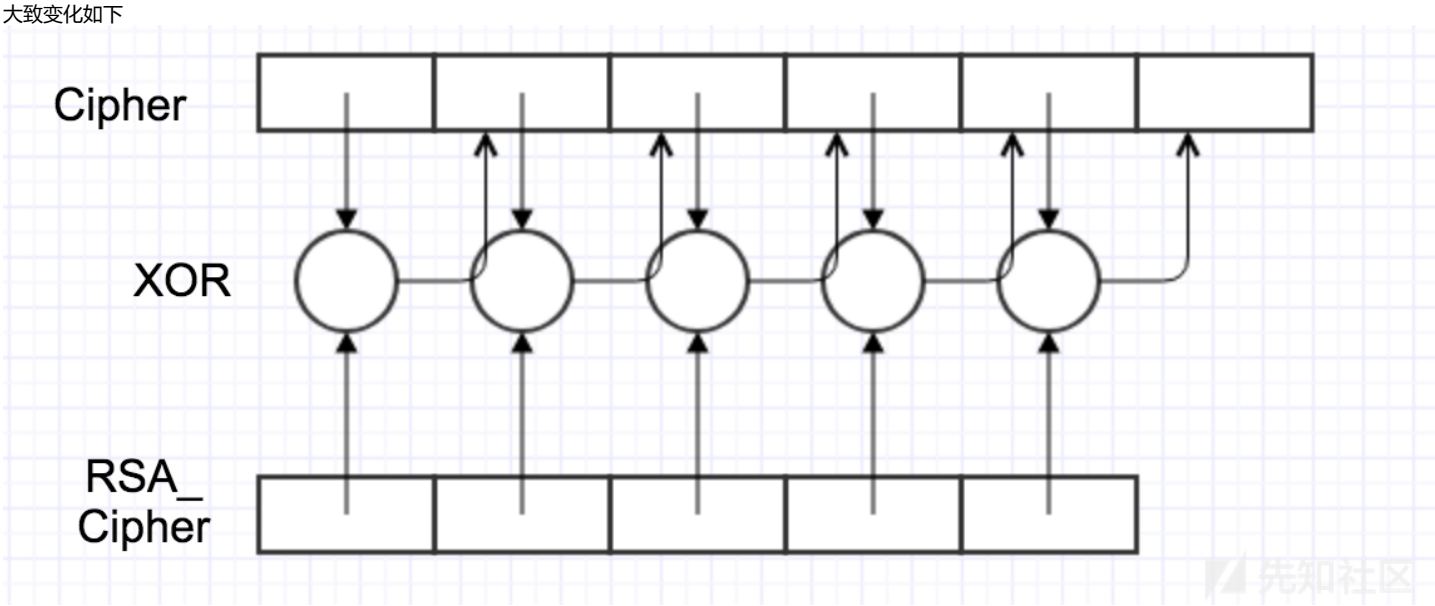
首先是padded

```
iv = os.urandom(256)
output = iv
hexencoded = binascii.hexlify(data)
length_target = (len(hexencoded) + 3) // 4
padded = hexencoded.ljust(length_target * 4, b'f')
```

保证data的hex长度是4的倍数，不够用f补齐
然后

```
for i in range(0, len(padded), 4):
       r = rsa_encrypt(padded[i:i+4])
       block = binascii.unhexlify('%.512x' % r)
       output += xor(output[-256:], block)
```

将之前加上padding的密文每4个一组，然后进行rsa加密，然后格式化为512长度的无符号16进制，再进行异或加密，这里异或的key是`output[-256:]`，并且一直在变化
大致变化如下



想反解也很容易
用C的后一块异或前一块即可
可以理解为

```
c[i-256:i]^c[i:i+256]
```

即可得到RSA的密文
那么这里的RSA怎么处理呢？
我们看一下RSA的n和e

```
p = number.getPrime(1024)
q = number.getPrime(1024)
n = p * q
e = number.getPrime(24)
```

这样随机生成的大n是无法直接破解的，但是我们不难发现
这里RSA加密的消息长度极短，长度只有4，所以这里我们可以用他的n和e遍历加密一遍
生成一个我们的字典，然后根据密文，选择明文即可
例如

```
dec = {}
for i in range(0x10000):
   x = b'%.4x' % i
   v = number.bytes_to_long(x)
   dec[pow(v, e, n)] = x
```

这样即可成功破解layer4

解题脚本

那么根据上述的思路，我们容易写出如下脚本

```
import base64
from Crypto.Util import number
n=0x80dd2dec6684d43bd8f2115c88717386b2053bdb554a12d52840380af48088b7f1f71c3d3840ef4615af318bbe261d2d2d90616c0d2dcb6414e05c706f
```

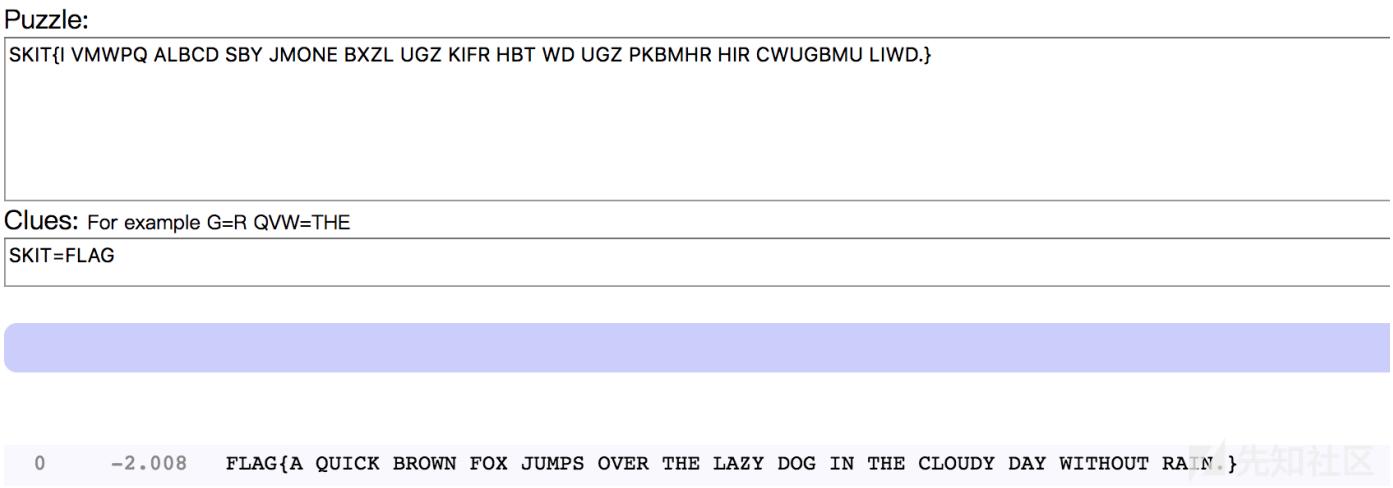
```
e=0xcf98d5
lines = open('encrypted').readlines()
data = base64.b64decode(lines[3].strip())
def xor(a, b):
   res=''
   for i in range(len(a)):
       res+=chr(ord(a[i])^ord(b[i]))
   return res
dec = {}
for i in range(0x10000):
   x = b'%.4x' % i
   v = number.bytes_to_long(x)
   dec[pow(v, e, n)] = x
raw = b''
for i in range(256, len(data), 256):
   prev = data[i-256:i]
   curr = int(xor(prev, data[i:i+256]).encode('hex'), 16)
   raw += dec[curr]
data = raw.decode('hex')
r = number.inverse(17, 251)
for key in range(0,256):
   output=''
   res=''
   for i in data:
       key = (key * 0xc8763 + 9487) % 0x10000000000000000
       output+=chr((ord(i) ^ key) & 0xff)
   for i in output:
       res += chr((ord(i)*r)%251)
   if res[4:5]=='{' and res[-2:] == '}\n':
       print res
       break
```

运行后得到

```
SKIT{I VMWPQ ALBCD SBY JMONE BXZL UGZ KIFR HBT WD UGZ PKBMHR HIR CWUGBMU LIWD.}
```

然后去

```
https://quipqiup.com/
```



即可得到flag

后记

从这道题里，可以学到一些代换的知识，并且可以得出一个不成熟的推论，就是密码题中如果频繁使用了xor，那么一定有问题XD

- 动动手指，沙发就是你的了！

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板