2018护网杯线上赛——Crypto——fez

转载

weixin_30378623 于 2018-10-16 19:53:00 发布 127 收藏
文章标签： python
原文链接：http://www.cnblogs.com/nldyy/p/9800260.html

注：本人较水，此题是看了众多大佬的writeup后才看明白的，写这个只是为了自己更好的理解。

题目：

fez.py、fez.log文件

fez.py文件

```
import os
def xor(a,b):
    assert len(a)==len(b)
    c=""
    for i in range(len(a)):
        c+=chr(ord(a[i])^ord(b[i]))
    return c
def f(x,k):
    return xor(xor(x,k),7)
def round(M,K):
    L=M[0:27]
    R=M[27:54]
    new_l=R
    new_r=xor(xor(R,L),K)
    return new_l+new_r
def fez(m,K):
    for i in K:
        m=round(m,i)
    return m

K=[]
for i in range(7):
    K.append(os.urandom(27))
m=open("flag","rb").read()
assert len(m)<54
m+=os.urandom(54-len(m))

test=os.urandom(54)
print test.encode("hex")
print fez(test,K).encode("hex")
print fez(m,K).encode("hex")
```

fez.log文件

```
51026a40ec1e8dbc84afe07fa1678629bd52dbbefc7037c2c38665401e066031b8120d687098b588f65aa09f974279bd3a8352adcd8
2ae39981788cf243a32ba7ba7c41b451d74f6b8941bb6222f92f011a1ffdffd8dff28487ba45dc0b88d3f249ec73aa660b13c2e7117
0c9be28d74519aa64d689d2de80063f51fdccc239fa6d6041f03240b098dd4437a72ae9933f36b49cbe314631b39881b0aff4af2dd5
```

分析：

知识点： 1.a^a=0;

2.a^a^b=b;

3.由前两个式子可以推出若：a^b=aa;c^b=cc;则有aa^cc=a^c;(本题重要的解题点)

首先，小白的我认真的读了代码，对python函数清一色靠百度。

代码大意：

1.K 是一个有7个元素的数组，每个元素是一个长度为27的字符串。

2.test是一个长度为54的随机字符串。

3.所以题目中加密方式一样，flag一样但每次所给的fez.log内容都不一样。

4.fez.log中的三行字符串分别是，test，test与k进行加密的结果，k与m(也就是flag)进行加密的结果。

5.每次加密都是调用fez()函数，在这个函数是循环7次，每次都是将K的元素与传入的另一个54位的字符串按round()函数那样进行运算，在round()函数中调用了xor()函数，xor()函数就是异或运算。

正面模拟一下这个fez.py的加密方式。

test和K。test的左边为Lt,右边为Rt。

第一次循环: Rt+Rt^Lt^K1

第二次循环: Rt^Lt^K1+Lt^K1^K2

第三次循环: Lt^K1^K2+Rt^K2^K3

第四次循环: Rt^K2^K3+Lt^Rt^K1^K3^K4

第五次循环: Lt^Rt^K1^K3^K4+Lt^K1^K2^K4^K5

第六次循环: Lt^K1^K2^K4^K5+Rt^K2^K3^K5^K6

第七次循环: Rt^K2^K3^K5^K6+Rt^Lt^K1^K3^K4^K6^K7……T_K

K和m。m的左边为Lm，右边为Rm。

第一次循环: Rm+Rm^Lm^K1

第二次循环: Rm^Lm^K1+Lm^K1^K2

第三次循环: Lm^K1^K2+Rm^K2^K3

第四次循环: Rm^K2^K3+Lm^Rm^K1^K3^K4

第五次循环: Lm^Rm^K1^K3^K4+Lm^K1^K2^K4^K5

第六次循环: Lm^K1^K2^K4^K5+Rm^K2^K3^K5^K6

第七次循环: Rm^K2^K3^K5^K6+Rm^Lm^K1^K3^K4^K6^K7……K_M

此时我们可以根据第3个知识点，发现，将这两次密文结果进行异或运算可以消除K，也就是密钥 。

即T_K^K_M=Rt^Rm+Rt^Lt^Rm^Lm……T^M

这时只要再将结果异或T就可以得到M：Rt^Rm^(Rt)=Rm

Rt^Lt^Rm^Lm^(Rt^Lt^Rm)=Lm

Lm+Rm=m即包含flag的字符串。

解题代码：

```
def xor(a,b):
    assert len(a)==len(b)
    c=""
    for i in range(len(a)):
        c+=chr(ord(a[i])^ord(b[i]))
    return c

test='51026a40ec1e8dbc84afe07fa1678629bd52dbbefc7037c2c38665401e066031b8120d687098b588f65aa09f974279bd3a8352adcd80'.decode('hex')
test_K='2ae39981788cf243a32ba7ba7c41b451d74f6b8941bb6222f92f011a1ffdffd8dff28487ba45dc0b88d3f249ec73aa660b13c2e71173'.decode('hex')
K_M='0c9be28d74519aa64d689d2de80063f51fdccc239fa6d6041f03240b098dd4437a72ae9933f36b49cbe314631b39881b0aff4af2dd5a'.decode('hex');

Lt=test[0:27]
Rt=test[27:54]

#Kl=K2^K3^K5^K6  Kr=K1^K3^K4^K6^K7
Kl=xor(test_K[0:27],Rt)
Kr=xor(Lt,xor(test_K[27:54],Rt))

Mr=xor(Kl,K_M[0:27])
Ml=xor(Mr,xor(Kr,K_M[27:54]))

print Ml,Mr
```

转载于:https://www.cnblogs.com/nldyy/p/9800260.html