WhaleCTF 密码学_Writeup

原创

Pad0y 于 2019-04-21 16:47:44 发布 2307 收藏 2

分类专栏： WhaleCTF

本文链接：https://blog.csdn.net/qq_34356800/article/details/85226207

期末事情有点多，新年放一波完整wp

Death_Chain

夏多密码，对照翻译即可，参考链接如下

https://www.33iq.com/group/topic/242254/

PS:所有的变换都要以原图中的为准，而不是基于上一次变换

先有什么

对着键盘画圈圈

检查符号

摩斯密码，写个替换脚本再翻译即可

```
# -*- coding: utf-8 -*-
# __author__: Pad0y

import re
s = "o00。o。o0oo。0o0o。000。00。o。0。000。ooo0。o。0o。oo0。ooo。0o0o。0。oo0o"
a = ["。", "o","0"]
b = [" ", ".","-"]
dic = dict(zip(a,b))
pattern = re.compile('(' + '|'.join(a) + ')')
s = pattern.sub(lambda a:dic[a.group()], s)
print(s)
```

德军密码

费娜姆密码，密文每7个一组，与密钥进行异或处理，详情看这里，脚本如下

```
# coding=utf-8
# __author__: Pad0y

import re


def b2ten(string):
    split = re.findall(r'.{7}', string)
    ten = []
    for i in split:
        ten.append(int(i, base=2))
    return ten

bin_t= '0000011000000000010101011011100101100010110000011100110010011110011100 1'
private_key= 'helloworld'
b2ascii = map(ord, private_key)
a_ten = b2ten(bin_t)
c = []
for i in range(len(a_ten)):
    c.append(a_ten[i] ^ b2ascii[i])
print ''.join(map(chr, c))
```

密钥生成

找个工具算下

规则很公平

tips是公平的游戏规则，想到了Playfair密码，也是一种古典密码，基于字符替换的密码。参考如下

Playfair算法是基于一个5*5的字母矩阵，
题目中CGOCPMOFEBMLUNISEOZY是密文，
关键词矩阵题目已经构造好了，比较省事。

- 矩阵构造规则

```
按从左到右、从上到下顺序
填入关键词的字母(去除重复字母)后
将字母表其作余字母填入
```

- 加密规则

```
Playfair加密算法是先将明文按两个字母一组进行分组，然后在矩阵中找对应的密文。

取密文的规则如下：
- 若明文出现相同字母在一组，则在重复的明文字母中插入
  一个填充字母(eg:z)进行分隔后重新分组(eg: balloon被重新分组为ba lz lo on)

- 若分组到最后一组时只有一个字母，则补充字母z
  若明文字母在矩阵中同行，则循环取其右边下一个字母为密文
  (矩阵最右边的下一个是最左边的第一个)(eg: ar被加密为RM)

- 若明文字母在矩阵中同列，则循环取其下边下一个字母为密文
  (矩阵最下边的下一个是最上边的第一个)(eg: mu被加密为CM)

- 若明文字母在矩阵中不同行不同列，则取其同行且与同组另一字
  母同列的字母为密文(eg: hs被加密为BP，ea被加密为IM或JM)
```

为了方便用numpy构造下5*5的关键词矩阵(其实是懒的画图)

```python
# coding=utf-8
# __author__: Pad0y

import numpy as np
s = 'CULTREABDFGHIKMNOPQSVWXYZ'
print(np.array(list(s)).reshape(5, 5))

"""
手动解密对照如下，python两行代码的事情（pycipher）

[['C' 'U' 'L' 'T' 'R']
 ['E' 'A' 'B' 'D' 'F']
 ['G' 'H' 'I' 'K' 'M']
 ['N' 'O' 'P' 'Q' 'S']
 ['V' 'W' 'X' 'Y' 'Z']]
 """
# 从矩阵可以看出密钥是CULTRE
# CG OC PM OF EB ML UN IS EO ZY
```

栅栏加密

根据题目描述是分成了3根栅栏，将密文每3个字符一组，将第一列与第二列换一下位置(用notepad/sublime的列编辑模式很容易做到)，得到如下

```
duJ
mZl
V2Y
uVW
dkx
XXs
N2e
D1V
V59
EXs
Z2d
7ZW
SlN
Vbr
9me
DNS
alF
GX9
F1
```

然后将所得到的字符串从上到下从左至右组合在一起再base64解码即可
numpy很容易就可以实现转换过程，解密脚本如下

```python
# -*- coding: utf-8 -*-
# __author__: Pad0y

import numpy as np
import base64

s = 'udJZml2VYVuWkdxXXs2Ne1DV5V9XEs2ZdZ7WlSNbVrm9eNDSlaFXG91F*'  # 字符串末尾补充*,构造19*3的矩阵
arr = np.array(list(s)).reshape(19, 3).T  # 矩阵转置
arr[[0, 1], :] = arr[[1, 0], :]  # 交换第一行和第二行的数据
f = arr.flat  # 数组扁平化
tmp = ''
for item in f:
    tmp += item
dec = base64.b64decode(tmp[:len(tmp)-1])  # 去掉补充的字符再解码
print(dec)
```

小明入侵

思路就是对管理员密码进行穷举加密，再和所给的部分md5比较，由于md5的特性若是前10位符合基本就是所得密码，爆破脚本如下：

```python
# -*- coding: utf-8 -*-
# __author__: Pad0y

import string
import hashlib

s = 'a74be8e20b'
chars = string.ascii_letters + string.digits  # 构造字符集
for i in chars:
    for j in chars:
        for k in chars:
            for n in chars:
                psw = 'key{' + i + j + k + n + '}'
                md5 = hashlib.md5(psw.encode(encoding='utf-8')).hexdigest()
                if s in md5:
                    print(psw, md5)
```

数学小问题

$$cipher = (ax+b) \bmod 26$$

这道题被坑的不惨，一开始没注意密文还夹杂着个1，看到图片很容易想到是仿射密码，但是仿射密码不可能存在数字。仿射密码m在这里限制为26，因此a的乘法逆元可能性只有12种，算上b偏移量26，密钥空间为12*26=312个，懒得写算法，直接调用解密网站接口爆破。为了不增加网站负担，只放部分源码，提供密钥K(5,8)

关于仿射密码详见此处

求a逆元函数如下

```python
# -*- coding: utf-8 -*-
# __author__: Pad0y

import math
from Crypto.Util.number import inverse


def rev_a(m):
    a = []  # a 与 26互模集合
    rev_a = []  # 逆元集合
    for i in range(1, m + 1):
        if math.gcd(i, m) == 1:
            a.append(i)
    for i in a:
        rev_a.append(inverse(i, m))
    return rev_a
```

此处应写

```
_____*((__//__+___+_____-____%____)**((___%(___-_))+_______+(___%___+_____+_____%__+_____-(_____//(_____%_
__)))))+__*(((_______/__)+___%__+_____-(_______//____))**(_*(_____+_____)+_____+_____%___))+_______*(
((_________//__+_______%__)+(_____-_))**((___+_____)+_____-(_____//__)))+_____*((___+_____-(____
__//___-_____%__%_))**(_____+_____+_____))+__*(__+_____-(___//___-_____%____%__))**(_____-____+__
_____)+(___+_____)**(_____%___%__+_____+_____)+(_____-__)*((____//____-____%____%_)+_____)**(_____-(_
_____//_____+_____%___)+_____)+(_____+(_____%_____)*__+_)**_____+_____*(((_____%_____)
*__+_____-(_____//_____))**_____)+(_____/__)*(((____-_+_____)*(_____+____))**___)+___*((__+_____
____-_)**_____)+___*(((___+_____-_____/___+__-_____%____%__)*(___-_+_____/__+_____%_____))**__)+(
_//_)*(((_____%___%__+_____+_____)%_____)+_____-_)**___+_____*((_____/(_____%___))+_____)*((_____%_
_____)*__+_____+_)+___//___+_____+_____/___
```

看起来是个数学填空题，口算是不存在的，上脚本

```python
# -*- coding: utf-8 -*-
# __author__: Pad0y

with open('txt', 'r') as f:
    s = f.read()
count = 0
exp = ''
for i in s:
    if i is s[0]:
        count += 1
    else:
        if count != 0:
            exp += str(count)
            count = 0
            exp += i
        else:
            exp += i

if count != 0:
    exp += str(count)
exp = exp.replace('//', '/')  # 文本中除号做转义需要去掉
print(exp + '\n' * 4, int(eval(exp)))

"""
提交结果不对折腾了半小时，丢到小葵做各种蜜汁转换，得到key
只需要转为16进制再转字符即可
"""
```

后来发现个更骚气的东西，下划线个数代表对应数字，不用爆破

```
import binascii
_ = 1
__ = 2
___ = 3
____ = 4
_____ = 5
______ = 6
_______ = 7
________ = 8
_________ = 9

a = _____*((__//__+___+______-____%____)**((___%(___-_))+________+(___%___+_____+_______%__+______-(______//(___
__%___)))))+__*(((________/__)+___%__+_______-(________//____))**(_*(_____+_____)+_______+________%___))+______
__*(((_________//__+________%__)+(_______-_))**((___+_______)+_________-(______//__)))+_______*((___+_________-(
______//___-_______%__%_))**(_____+_____+_____))+__*(__+_________-(___//___-________%_____%__))**(_________-___
_+_______)+(___+_______)**(________%___%__+_____+______)+(_____-__)*((____//____-_____%____%_)+_________)**(____
_-(_______//_______+________%___)+______)+(_____+(________%_______)*__+_)**________+_______*(((________%____
___)*__+_______-(________//________))**_______)+(________/__)*(((____-_+_______)*(______+____))**___)+___*((__+_
_______-_)**_____)+___*(((___+_______-______/___+__-________%_____%__)*(___-_+_______/__+________%_____))**_
_)+(_//_)*(((________%___%__+_____+_____)%______)+_______-_)**___+_____*((_____/(_____%___))+_______)*((_______
__%_______)*__+_____+_)+___//___+________+________/___
a = hex(a)[2:][:-1]
a = binascii.a2b_hex(a)
print a
```

RSA分解

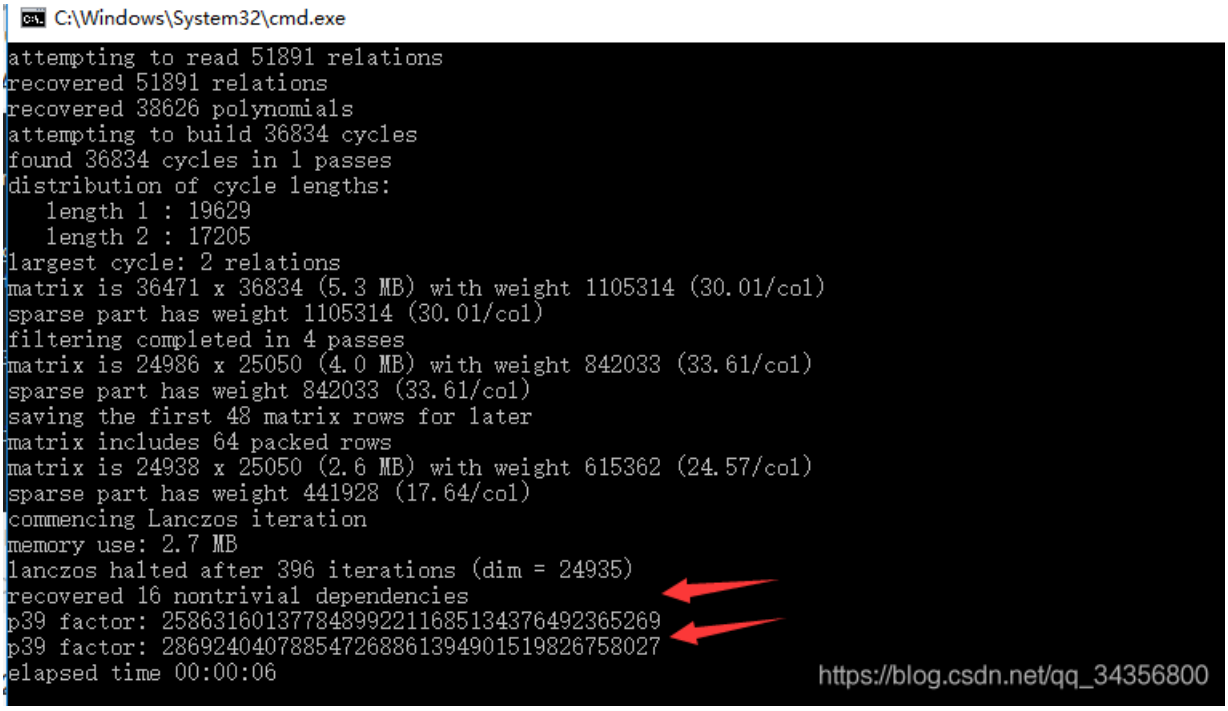
解压得到密文和公钥，对公钥解析得到e = 65537(0x10001)
n=0xA41006DEFD378B7395B4E2EB1EC9BF56A61CD9C3B5A0A73528521EEB2FB817A7

```
C:\Users\Pamper\Desktop
λ openssl.exe rsa -pubin -text -modulus -in warmup -in public.pem
Public-Key: (256 bit)
Modulus:
    00:a4:10:06:de:fd:37:8b:73:95:b4:e2:eb:1e:c9:
    bf:56:a6:1c:d9:c3:b5:a0:a7:35:28:52:1e:eb:2f:
    b8:17:a7
Exponent: 65537 (0x10001)
Modulus=A41006DEFD378B7395B4E2EB1EC9BF56A61CD9C3B5A0A73528521EEB2FB817A7
writing RSA key
-----BEGIN PUBLIC KEY-----
MDwwDQYJKoZIhvcNAQEBBQADKwAwKAIhAKQQBt79N4tzlbTi6x7Jv1amHNnDtaCn
NShSHusvuBenAgMBAAE=
-----END PUBLIC KEY-----
```

用msieve（yafu也行）分解n得到

p = 258631601377848992211685134376492365269

q = 286924040788547268861394901519826758027



脚本如下

RSA分解

- 解题思路：给出了公钥对对密文进行遍历解密即可

```
N = 920139713 E = 19,分解N，得到

fac: factoring 920139713
fac: using pretesting plan: normal
fac: no tune info: using qs/gnfs crossover of 95 digits
div: primes less than 10000
fmt: 1000000 iterations
Total factoring time = 0.0421 seconds
***factors found***
P5 = 49891
P5 = 18443
ans = 1
```

可求出d=96849619 所以私钥为(920139713,96849619)

```python
# -*- coding: UTF-8 -*-
# __author__:Pad0y
n = 920139713
d = 96849619
result = []
with open("rsa.txt") as f:
    for i in f:
        result.append(chr(pow(int(i),d,n)))
print(result)
```

只有密文

- 唯密文攻击

思路：计算q=n/p,对比找出小的那个质数，找到n与200组密文的最大公约数即相当于对n做了分解

1. **把文本里第一个数据去掉，剩下200个密文，计算p**

```python
# -*- coding: UTF-8 -*-
# __author__:Pad0y

import re

with open('ciphertext.txt', 'r') as f:
    content = f.readlines()
    e = []
    for line in content:
        res = re.findall('^\d+', line)
        if len(res) > 0:
            e.append(int(res[0]))

n = 135176830582884945708175419898330054260341730432046991449072509302750602166218145078102928897914789996197402
658592881347572949256377161172079344803330624352445165759925647345536051853372740246104804540179716136644319380454884518397455488002758429914465640804944658049262500561494830899678619427468784748988379


def divisors(m, n):
    c = 1
    while c != 0:
        c = m % n
        m = n
        n = c
    return m


if __name__ == '__main__':
    for i in range(len(e)):
        print(str(i + 1) + ':' + str(divisors(n, e[i])))
```

最后只有第96组和n最大公约数不是1，即
p=13038371855775914836995578093728166671103633520203033965827703187246607207039273968425501296569317295959057439253867586769212037981452712871242668046329877

2. **计算q**

q=n/p，得到
q=10367615840240242845371941453623373821227053765532752994306127876946421006862147600725324340607889088707606730457021312059130583835286311559997627141422127

3. 比较大小，得到q比较小，进行md5加密

```
# -*- coding: UTF-8 -*-
# __author__: Pad0y

import hashlib

q = 10367615840240242845371941453623373821227053765327529943061278769464210068621476007253243406078890887076067
30457021312059130583835286311559997627141422127
m = hashlib.md5()
m.update(str(q).encode('ascii'))
enc = m.hexdigest()
print('key{' + enc[:8] + '}')
```

算法问题

运行下发现结果和给的enc文本一样，果断把源码上的flag丢上去(捂脸)

大家来解密

CBC模式的AES加密，AES共有五种加密模式（ECB，CBC，PCBC，CFB，OFB，CTR），其中CBC是公认最安全的模式
KEY=venusCTF-hex IV=123-MD5代表分别把key和偏移量转化为后面格式
IV在md5后长度是32位，需要对key填充之相应位数再编码

```
# -*- coding: utf-8 -*-
# __author__: Pad0y

import binascii
import hashlib
from Crypto.Cipher import AES

KEY = b'venusCTF'
IV = '123'

iv = hashlib.md5(IV.encode(encoding='utf-8')).hexdigest()
key = (KEY * 4).hex()
c = 'a80d5eb43508e549f83e2e254c0a0f0644be58f453baced4af4777c4cd1b7575'

k = binascii.unhexlify('76656e7573435446766656e7573435446766656e7573435446766656e7573435446')
key_ = binascii.unhexlify(key)
iv_ = binascii.unhexlify(iv)
C = binascii.unhexlify(c)
aes = AES.new(key_, AES.MODE_CBC, iv_)
print(aes.decrypt(C))
```

RSA专家

得到私钥和密文，丢到openssl

```
C:\Users\Pamper\Desktop\rsa
λ openssl.exe rsautl -in endata -out decdata.txt -inkey pri_k -decrypt -pkcs
```

AK

密码学

Death_Chain 100	先有什么 100	检查符号 100	德军密码 100
密钥生成 100	规则很公平 150	数学小问题 150	此处应写 150
栅栏加密 150	小明入侵 150	RSA破解 200	RSA分解 200
只有密文 200	算法问题 200	大家来解密 200	RSA专家 200