20211107-陇原战疫-Crypto方向WP

原创

4XWi11 于 2021-11-08 16:31:23 发布 5318 收藏 1

分类专栏： 树哥让我天天写之Crypto 文章标签： python

本文链接：https://blog.csdn.net/m0_49109277/article/details/121210550

树哥让我天天写之Crypto 专栏收录该内容

21 篇文章 5 订阅

订阅专栏

原文地址

https://4xwi11.github.io/posts/a1174b3b/

陇原战"疫"2021网络安全大赛

Crypto

mostlycommon

共模再开个方

```python
import gmpy2
from Crypto.Util.number import *

e1 = 65536 // 2
e2 = 270270 // 2

n = 12203168613869661959991469076776428609456284211208822531150382601400688603906908319297459971268502782511168485223523003918221624502971478648054108710508189533925140373870336939955159388293189639250083206107041448323302906711741095249965548216010402773046274049734721275226958952626750410026270736702024461350 3
c1 = 39449016403735405892343507200740098477581039605979603484774347714381635211925585924812727991400278031892391996192354880233130336052873275920425836986816735715003772614138146640312241166362203750473990403841789871473337067450727600486330723461100602952736232306602481565348834811292749547240619400084712149673
c2 = 43941404835820273964142098782061043522125350280729366116311943171108689108114444447295511969090107129530187119024651382804933594308335681000311125969011096172605146903018110328309963467134604392943061014968838406604211996322468276744714063735786505249416708394394169324315945145477883438003569372460172268277

get_tuple = gmpy2.gcdext(e1, e2)        # return (g,s,t)   g = gcd(e1,e2) and g = e1*s + e2*t

r = -get_tuple[1]
s = get_tuple[2]

a = gmpy2.powmod(gmpy2.invert(c1, n), r, n)
b = gmpy2.powmod(c2, s, n)

m = gmpy2.f_mod(gmpy2.mul(a,b),n)
print(gmpy2.iroot(m, 2))
print(long_to_bytes(gmpy2.iroot(m, 2)[0]))
```

easytask（recuring）

差点以为是和省赛NTRU那样的题目（其实就是

虽然`r`是完全可以爆破的，但是要3个小时才能跑完（再买个内存条

所以可还行，emmmm，写了个脚本但比赛中没有出

有能力出，但没出，总结一下几点

1. 遍历所有的`r`不慢，但是不精明的操作（在循环中生成AES解密）会导致空间复杂度很大，导致程序非常慢
2. 赛后问尚师傅，其实真正的`r`开头是`-3 -2`（应该是刻意为之），所以尽管空间复杂度很大，但是比赛中还是遍历到了，判断机制写得不对（马虎，还有别的原因
3. 综上，判断机制不是非得是`flag.starswith(b'flag')`，求出来的`m`很有特征的；可以学尚师傅，先自己定个`r`，然后用脚本跑一遍（处理大数据，取样显然效率更高
4. 比赛中先算出`r`保存在内存里，再开始遍历

调整后的脚本快多了

```
#!/usr/bin/env sage
# -*- coding: utf-8 -*-
import hashlib
from Crypto.Cipher import AES
import re
from itertools import product

c = '1070260d8986d5e3c4b7e672a6f1ef2c185c7fff682f99cc4a8e49cfce168aa0'
c = bytes.fromhex(c)
ct_e = '[151991736758354 115130361237591  58905390613532 130965235357066  74614897867998  48099459442369  458944
85782943   7933340009592     25794185638]'
ct_W = '''[-10150241248 -11679953514  -8802490385 -12260198788 -10290571893   -334269043 -11669932300  -21588274
58     -7021995]
[ 52255960212  48054224859  28230779201  43264260760  20836572799   8191198018  14000400181   4370731005     142
51110]
[  2274129180  -1678741826  -1009050115   1858488045    978763435   4717368685   -561197285  -1999440633     -65
40190]
[ 45454841384  34351838833  19058600591  39744104894  21481706222  14785555279  13193105539   2306952916      75
01297]
[-16804706629 -13041485360  -8292982763 -16801260566  -9211427035  -4808377155  -6530124040  -2572433293     -83
93737]
[ 28223439540  19293284310   5217202426  27179839904  23182044384  10788207024  18495479452   4007452688     130
46387]
[   968256091  -1507028552   1677187853   8685590653   9696793863   2942265602  10534454095   2668834317      86
94828]
[ 33556338459  26577210571  16558795385  28327066095  10684900266   9113388576   2446282316   -173705548      -5
77070]
[ 35404775180  32321129676  15071970630  24947264815  14402999486   5857384379  10620159241   2408185012      78
41686]'''

e = []
for i in re.findall(r"\d+", ct_e):
    e.append(int(i))
e = matrix(e)

W = [[0 for _ in range(9)] for j in range(9)]
ct_W = re.findall(r'-?\d+', ct_W)
for i in range(len(ct_W)):
    W[i // 9][i % 9] = int(ct_W[i])
W = matrix(W)
inv_W = W.inverse()

table = [_ for _ in range(-3, 4)]
r = product(table, repeat=9)

for ri in r:
    rx = matrix(ri)
    m = (e - rx) * inv_W
    if 0 < m[0][0] < 1024:
        M = list(m[0])
        key = hashlib.sha256(str(M).encode()).digest()
        cipher = AES.new(key, AES.MODE_ECB)
        flag = cipher.decrypt(c)
        if flag.startswith(b'flag') or flag.startswith(b'SET'):
            print(r)
            print(flag)
            break
```

让我看看之前的脚本为啥不能出

我们自己写一个`r`，就`[-3,-3,-3,-3,-3,-3]`吧，然后走一遍，我知道了，是这个问题

```
M = list(m[0])
```

sage矩阵转列表的时候要多取一个，差不多是这么一个意思吧

```
sage: m = [-3 for _ in range(9)]
sage: m
[-3, -3, -3, -3, -3, -3, -3, -3, -3]
sage: m = matrix(m)
sage: m
[-3 -3 -3 -3 -3 -3 -3 -3 -3]
sage: m = list(m)
sage: m
[(-3, -3, -3, -3, -3, -3, -3, -3, -3)]
sage: m[0]
(-3, -3, -3, -3, -3, -3, -3, -3, -3)
```

寄

Civet cat for Prince（recuring)

代码小饶

目的，已知IV，可以获得`输入name+'a_cat_permission'`的密文，可以有最多两次自己输入IV和密文进行解密，求`输入name+'Princepermission'`的密文，**注意最后这里我们可以自己定义IV**

重点是最后一句话，这就是利用点，复现的时候我一个上午都没注意到，以为是要用它的IV；然后下午看到，那就比较简单了

一个小思路，假设获取的`name_cipher`c分成前半段

c_1c_2，密文构造成

然后利用一次解密的机会，将上面这个密文的前半段和系统的IV丢进去，得到的结果记为m，求

$$\oplus IV \oplus Princepermission$$

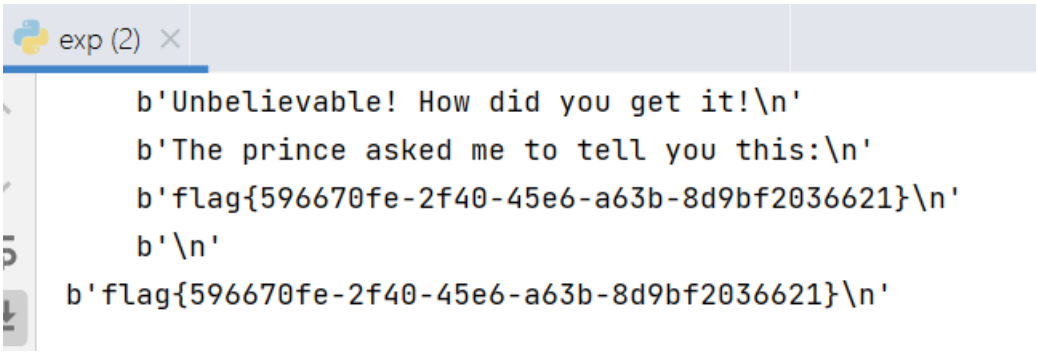
到此这道题已经被我们攻破，可能有点绕，但是海星

写了个jo本，源文件改过了？打本地可以，远程有时候可以，可能是有些结果有换行

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import hashlib
from pwn import *
from itertools import product
import string

context.log_level = 'debug'
table = string.ascii_letters + string.digits


class Solve:
    def __init__(self):
        # self.sh = remote('node4.buuoj.cn', 28742)
```

```
    # self.sh = remote('node4.buuoj.cn', 28743)
    self.sh = remote('node4.buuoj.cn', 29129)
    self._Princepermission = b'Princepermission'
    self._a_cat_permission = b'a_cat_permission'
    self.iv = b''
    self.cipher_name = b''
    self.payload_cipher = b''
    self.payload_iv = b''

def proof_of_work(self):
    """
    [+] sha256(XXXX+Q3kqSv2c) == e9ded46c9d0dbcf14d8c36852678fe59daec43b1025c282738a81e7ea9f395f9
    [+] Give Me XXXX :
    """
    proof = self.sh.recvline()
    tail = proof[16:24].decode()
    HASH = proof[29:93].decode()
    for i in product(table, repeat=4):
        head = ''.join(i)
        t = hashlib.sha256((head + tail).encode()).hexdigest()
        if t == HASH:
            self.sh.recvuntil(b'[+] Give Me XXXX :')
            self.sh.sendline(head.encode())
            break

def solve_BANNER(self, _name):
    self.sh.sendlineafter(b'[-]', b'1')
    self.sh.sendlineafter(b'[-]', _name)
    self.sh.recvline()
    self.sh.recvline()
    self.sh.recvuntil(b'Miao~ ')
    self.iv = self.sh.recvuntil(b"I'm a")[:-6]
    print(len(self.iv))

def solve_NAME(self):
    self.sh.sendlineafter(b'[-]', b'1')
    self.sh.recvuntil(b'Permission:')
    self.cipher_name = self.sh.recvuntil(b"I'm a")[:-6]
    self.cipher_name = self.cipher_name
    self.payload_cipher = xor(xor(self.cipher_name[:16], self._a_cat_permission),
                              self._Princepermission) + self.cipher_name[16:32]

def solve_Princepermission(self):
    self.sh.sendlineafter(b'[-]', b'2')
    self.sh.sendlineafter(b'[-]', self.payload_cipher[:16])
    self.sh.sendlineafter(b'[-]', self.iv)
    self.sh.recvuntil(b'The message is ')
    self.payload_iv = xor(xor(self.sh.recvuntil(b'1.getpermission')[:16], self.iv), self._Princepermission)

def solve_flag(self):
    self.sh.sendlineafter(b'[-]', self.payload_cipher)
    self.sh.sendlineafter(b'[-]', self.payload_iv)
    self.sh.recvuntil(b'The prince asked me to tell you this:\n')
    flag = self.sh.recvline()
    print(flag)

def solve(self):
    self.proof_of_work()
    self.solve_BANNER(self._Princepermission)
```

```python
        # get cipher_name
        # chance ====> 2
        self.solve_NAME()

        # get cipher_Princepermission
        # chance ====> 1
        self.solve_Princepermission()

        # chance ====> 0
        self.sh.sendlineafter(b'[-]', b'3')
        # get flag
        self.solve_flag()


if __name__ == '__main__':
    solution = Solve()
    solution.solve()
```



好一个狸猫换太子

Re

EasyRe

反编译失败，查了好久，尝试平衡栈，手动修无果，然后想起一开始看到一串32位的字符串没用到，随便搜了下竟然是`hello world`的md5，抱着试一试的心态丢进去就对了