Crypto

狂飙

```
import os
from flag import flag
from Crypto.Util.number import *
from Crypto.Cipher import AES

m = 880075137024242437020664908495968173048278395470076415264335977888002120652 49
key = os.urandom(24)
key = bytes_to_long(key)
n=m % key
# n=>key
m % key = n

flag += (16 - len(flag) % 16) * b'\x00'
iv = os.urandom(16)  # 8byte异或
aes = AES.new(key,AES.MODE_CBC,iv)
enc_flag = aes.encrypt(flag)

print(n)
print(enc_flag)
print(iv)


#10356084300607870894483365833917289619238951362 5588
#b'\xfc\x87\xcb\x8e\x9d\x1a\x17\x86\xd9~\x16)\xbfU\x98D\xfe\x8f\xde\x9c\xb0\xd1\x9e\xe7\xa7\xefiY\x95C\x14\x13C@j1\x9d\x08\xd9\xe7w>F2\x96cm\xeb'
#b'UN\x1d\xe2r<\x1db\x00\xdb\x9a\x84\x1e\x82\xf0\x86'
```

AES解密：创建AES解密器，调用`decrypt`方法

```
aes_dec = AES.new(key, AES.MODE_CBC, iv)
dec = aes_dec.decrypt(enc_flag)
```

补充：os.urandom() 函数生成的字节串是不可打印的，只能用于加密、哈希、签名等应用。在使用 os.urandom() 函数生成随机数时，可以将生成的字节串转换为整数，然后根据需要进行取模等操作。

```
from Crypto.Cipher import AES
import os

enc_flag =
iv = b'UN\x1d\xe2r<\x1db\x00\xdb\x9a\x84\x1e\x82\xf0\x86'
# print(iv)
max_num = 100
rand_int = int.from_bytes(iv, byteorder='big') % max_num + 1
print(rand_int)
```

言归正传，这道题`已知iv,cipher`，所以解密`关键在key`

- key:是24位随机数

- key是4位随机数
- `n=m % key`
- 已知m,n

思路：key是m-n的一个因子，通过大素数分解可以求得所有可能http://www.factordb.com/index.php

```
8800751370...61<77> = 3 · 37 · 439 · 3939851 · 265898280367<12> ·
5036645362649<13> · 342291058100503482469327892079792475478873<42>
```

通过`combinations`得到`因子组合成key的所有可能`，再逐个尝试解密

```
from Crypto.Cipher import AES
from itertools import combinations
from Crypto.Util.number import *

n = 103560843006078708944833658339172896192389513625588
m = 880075137024242437020664908495968173048278395470076415264335977888002120652 49
enc_flag =
b'\xfc\x87\xcb\x8e\x9d\x1a\x17\x86\xd9~\x16)\xbfU\x98D\xfe\x8f\xde\x9c\xb0\xd1\x9 e\xe7\xa7\xefiY\x95C\x14\x13C@j1\x9d\xO8\xd9\xe7W>F2\x96cm\xeb'
iv = b'UN\x1d\xe2r<\x1db\xOO\xdb\x9a\x84\x1e\x82\xf0\x86'

factor = [3, 37, 439, 3939851, 265898280367, 5036645362649,
342291058100503482469327892079792475478873]
i = 0
j = 0
k = 0
l = 0
s = 0
o = 0
num = 0
key_cand = []


def find_key(factor, n, m):
    for r in range(1, len(factor) + 1):
        # 生成所有可能的因子组合序列
        for comb in combinations(factor, r):
            key = 1
            for num in comb:
                # 每个组合内所有因子乘积为一个大因子
                key *= num
                # 除数大于余数
                if key > n:
                    if n == m % key:
                        key_cand.append(key)


key = find_key(factor, n, m)
for key in key_cand:
    key= long_to_bytes(key)
    try:
        aes_dec = AES.new(key, AES.MODE_CBC, iv)
        dec = aes_dec.decrypt(enc_flag)
        dec = dec.rstrip(b'\xOO')
```

```
        print(dec)
    except Exception as e:
        continue
```

官方题解：通过遍历所有除数获得key的所有可能，`long_to_bytes(i,24)`强制获得24位密钥，就不需要错误处理的环节了，最后`if ... in ...`找flag

```
from Crypto.Cipher import AES
from Crypto.Util.number import *
n=103560843006078708944833658339172896192389513625588
m=880075137024242437020664908495968173048278395470076415264335977888002120652 49
key=m-n
#880075137024242437020664907460359742987491306021739831872607015964106984396 61
enc=b'\xfc\x87\xcb\x8e\x9d\x1a\x17\x86\xd9~\x16)\xbfU\x98D\xfe\x8f\xde\x9c\xb0\xd 1\x9e\xe7\xa7\xefiY\x95C\x14\x13C@j1\x9d\xO8\xd9\xe7W>F2\x96cm\xeb'
iv=b'UN\x1d\xe2r<\x1db\xOO\xdb\x9a\x84\x1e\x82\xf0\x86'
for i in key.divisors():
    i=long_to_bytes(i,24)
    aes=AES.new(i,AES.MODE_CBC,iv)
    flag=aes.decrypt(enc)
    if b'flag{' in flag:
        print(flag)
```

MISC

checkin

提示：T-Rex Run!

```
Runner.prototype.gameOver = function(){};
Runner.instance_.distanceRan = 114514 /
Runner.instance_.distanceMeter.config.COEFFICIENT
```

PWN

二进制安全学习路线

ezhp_code

Heap overflow堆溢出（一）

An easy HeapOverflow