SCTF_2019_crypto_warmup

原创

bigbigliang 于 2019-06-26 08:56:38 发布 193 收藏
分类专栏： crypto 文章标签： ctf

初看本题，是个AES——CBC加密，key，iv都是随机生成的，研究了半天AES——CBC碰撞过程，没解出来。感觉跟之前的AES加密过程不太一样，之前的题，一般会给一个key或者iv，这次不一样。

过两天看了别人的writeup，发现的确不是正常解AES，然后通过xor绕过去的。

题目一开始提供了message='see you at three o\'clock tomorrow' 的AES计算结果。那么本题就是通过构造输入msg通过xor产生与message一样的结果。

题目中一共有两个主要的判断：
1.self.code(msg) == code 这是第一步要解决的

2.msg = self.unpad(msg)这是第二步要解决的

对于第一步：

对于传入的msg进行code加密，code函数里首先对输入的msg进行每16字节的xor，这里是关键，有个想法，可以通过构造msg使得xor结果与message的xor结果一样，这样aes加密后产生的结果就是一样的了。可以绕过第一步了。

对于第二步：

if msg == 'please send me your flag': 这里要求你输入的msg等于这个。在这之前还有一步

msg = self.unpad(msg) 在msg输入的末尾构造一个数，这里是'H'，通过unpad可以只保留前24个字节，在这里就是'please send me your flag'，可以成功进入第二步，拿到flag。

exp：

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Crypto.Cipher import AES
from Crypto.Util.strxor import strxor
from binascii import hexlify, unhexlify
from Crypto.Random import get_random_bytes
#from FLAG import flag
class MAC:
    def __init__(self):  #运行一次程序  key跟iv是一定的
        self.key = get_random_bytes(16)
        self.iv = get_random_bytes(16)
```

```
    def pad(self, msg):
        pad_length = 16 - len(msg) % 16
        return msg + chr(pad_length) * pad_length
    def unpad(self, msg):
        return msg[:-ord(msg[-1])]
    def code(self, msg):
        res = chr(0)*16
        for i in range(len(msg)/16):
            res = strxor(msg[i*16:(i+1)*16], res)     #xor           #res 已知
        print res          #res是异或后的明文
        aes = AES.new(self.key, AES.MODE_CBC, self.iv)
        return aes.encrypt(res).encode('hex')   #res已知   key iv 不定    不定50bd26086d6e3e0579ab0bdc55813de
    def identity(self, msg, code):
        print msg
        if self.code(msg) == code:   #第一步要解决
            msg = self.unpad(msg)
            print msg
            if msg == 'please send me your flag':
                print 'remote: ok, here is your flag:%s' % "hha"
            else:
                print 'remote: I got it'
        else:
            print 'remote: hacker!'
if __name__ == '__main__':
    mac = MAC()
    message = 'see you at three o\'clock tomorrow'
    #message ='please send me your flag'
    print 'you seem to have intercepted something:{%s:%s}' %(mac.pad(message).encode('hex'), mac.code(mac.p

    msg_write = mac.pad(message)   #48
    mac_write = b'\x00' * 16
    for i in range(len(msg_write) // 16):
        mac_write = strxor(msg_write[i * 16:(i + 1) * 16], mac_write)
    forge_msg = bytearray(b'please send me your flag' + (b'\x00' * 8))
    #print len(forge_msg)   #32
    forge_msg.extend(forge_msg)
    #print len(forge_msg)    #64
    forge_msg.extend(bytearray(mac_write))
    #print len(forge_msg)    #80
    length = len(forge_msg) + len(mac_write) - len('please send me your flag')  #80+16-24=72
    #print forge_msg[-1]    #5
    forge_msg[-1] ^= length
    forge_msg.extend(b'\x00' * 15)
    #print len(forge_msg)   #95
    forge_msg.append(length)
    print forge_msg    #96

    print 'so send your message:'
    #msg = raw_input()  # 16进制      706c656173652073656e64206d6520796f757220666c6167
    msg = hexlify(forge_msg)

    print 'and your code:'
    #code = raw_input()   #16进制
    code = mac.code(mac.pad(message))
    mac.identity(msg.decode('hex'), code)
    exit()
```