SCTF2019 Crypto-warmup writeup

转载

放错位的天才 于 2019-06-27 01:23:00 发布 204 收藏
文章标签： python
原文链接：http://www.cnblogs.com/KRDecad3/p/11094774.html

题外话

其实这道题在比赛过程中并没有解出来，思路完全想偏导致无解就放弃了，后来研究了大佬的writeup大半天才看懂。。。

正文

nc获取题目信息，返回一段明文和密文，要求输入一段明文和密文。
题目源码：

```
# server.py
#!/usr/bin/python
# -*- coding: utf-8 -*-

from Crypto.Cipher import AES
from Crypto.Util.strxor import strxor
from Crypto.Random import get_random_bytes
from FLAG import flag

class MAC:
    def __init__(self):
        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)
        aes = AES.new(self.key, AES.MODE_CBC, self.iv)
        return aes.encrypt(res).encode('hex')

    def identity(self, msg, code):
        if self.code(msg) == code:
            msg = self.unpad(msg)
            if msg == 'please send me your flag':
                print 'remote: ok, here is your flag:%s' % flag
            else:
                print 'remote: I got it'
        else:
            print 'remote: hacker!'


if __name__ == '__main__':
    mac = MAC()
    message = 'see you at three o\'clock tomorrow'
    print 'you seem to have intercepted something:{%s:%s}' %(mac.pad(message).encode('hex'), mac.code(mac
    print 'so send your message:'
    msg = raw_input()
    print 'and your code:'
    code = raw_input()
    mac.identity(msg.decode('hex'), code)
    exit()
```

通过identity函数可知，自己输入的明文在服务端加密后要等于自己输入的密文（也就是self.code(msg) == code，才能得到flag。

同时题目的坑点（也是我思路想歪的地方）就在这地方， 因为要求你输入的明文加密后等于你输入的密文，同时,加密使用的AES的CBC模式，对你的明文的加密是使用的和返回的第一段明文密文相同的iv和key，因此自然想到要得到iv和key才能求出要输入的密文。但是iv和key是通过随机数生成的，所以就无法用这个方法。

那么换一种思路，我们看看加密函数code如何工作的：

```
def code(self, msg):
       res = chr(0)*16
       for i in range(len(msg)/16):
           res = strxor(msg[i*16:(i+1)*16], res)
       aes = AES.new(self.key, AES.MODE_CBC, self.iv)
       return aes.encrypt(res).encode('hex')
```

既然AES无法破解，那么先看一下送进AES加密前明文，是将原明文分为每16位一组然后接连异或得到的16位字符串res，并且这个res我们可以求出来，也就相当于知道“明文”了。

如果我们直接使用第一段返回的'code'作为自己输入的'code'，那么我们只需要构造一串明文，使其通过code函数中AES加密前的操作，得到的结果，等于第一段的'res'，这样输入到服务端加密后的结果也等于了输入的'code'（即第一段code）。

好了，现在就开始考虑如何构造明文。

为了得到flag，我们倒着推明文，看identity函数，满足msg == 'please send me your flag'才能得到flag，其上一步是msg = self.unpad(msg)要求输入的明文脱去填充后的字符串是“please send me your flag”，看一下unpad函数，msg[:-ord(msg[-1])]和填充函数pad是相反的作用，即取末尾字符的ASCII码表示的范围之前的字符串。

现在我们将“please send me your flag”的长度记作len1 == 24，由于AES加密的要求，字符串填充后的长度要满足为16的倍数，因此这里可填充8+16n个字符（n=0,1,2...）。根据unpad函数，必须让填充后的字符串最末尾一个字符的ASCII码可表示填充的位数。

先跑一下code函数中的一部分（就是AES加密前），得出第一段密文前一阶段的'res'：

```
message = '73656520796f752061742074687265206f27636c6f636b20746f6d6f72726f770f0f0f0f0f0f0f0f0f0f0f0f0f'
for i in range(len(msg)/16):
    res = strxor(msg[i*16:(i+1)*16], res)
print(res.encode('hex'))
# res结果为24054d4c1a0f19444e0f4016080f1805
```

目标就是我们填充后的字符串异或处理后与上面的res（这里记作res1）相等。

我们设“please send me your flag”加上8位填充的32位并进行16位分组后得到的两个16位字符串为m1和m2（暂时先不管填充是什么）。由异或运算的性质可知，a XOR b XOR b = a。设m3为一个能使m1 XOR m2 XOR m3 XOR m4== res1的16位分组，由此可知m3 == res1 XOR m1 XOR m2 XOR m4。同时要满足字符串最末尾一个字符的ASCII码可表示填充的位数，并且此时字符串总长度位48，因此就再加一个16位m4分组保证最后一位表示填充长度（这里是40）。
那么如何得到m3呢，就是用code函数里的异或方式。

到此我们就完全知道要构造的输入明文了。

下面是解题的完整代码：

```
from Crypto.Cipher import AES
from Crypto.Util.strxor import strxor
from Crypto.Random import get_random_bytes
```

```
flag = 'test'

class MAC:
    def __init__(self):
        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)
        print(res.encode('hex'))  # 输出res1
        aes = AES.new(self.key, AES.MODE_CBC, self.iv)

        return aes.encrypt(res).encode('hex')

    def identity(self, msg, code):
        if self.code(msg) == code:
            msg = self.unpad(msg)
            if msg == 'please send me your flag':
                print 'remote: ok, here is your flag:%s' % flag
            else:
                print 'remote: I got it'
        else:
            print 'remote: hacker!'

if __name__ == '__main__':
    mac = MAC()
    message = 'see you at three o\'clock tomorrow'
    print 'you seem to have intercepted something:{%s:%s}' %(mac.pad(message).encode('hex'), mac.code(mac
    print 'so send your message:'
    msg = 'please send me your flag'
    # 使用和pad函数一样的填充方式先构成64位
    msg_p = msg + chr(64 - len(msg)) * (64 - len(msg))
    # 生成m1 XOR m2 XOR m4
    res = chr(0)*16
    for i in range(len(msg_p)/16 - 1):
        res = strxor(msg_p[i*16:(i+1)*16], res)
    # 最终输入的明文字符串
    # strxor("24054d4c1a0f19444e0f4016080f1805".decode('hex'), res) 即为上文的m3
    msg_p = msg_p[:32] + strxor("24054d4c1a0f19444e0f4016080f1805".decode('hex'), res) + msg_p[32:38]
    # 输出明文串
    print(msg_p.encode('hex'))
    print 'and your code:'
    code = raw_input()
    mac.identity(msg_p.encode('hex').decode('hex'), code)
    exit()
```

将得到的明文字符串输入，再将服务端返回的密文输入，就得到flag了。

参考

我主要参考这篇文章写出来的，所以思路也大都汲取自这位大佬的：
SCTF2019 部分题目WriteUp
搞懂之后感觉这道题真是妙啊（虽然我没做出来）。

转载于:https://www.cnblogs.com/KRDecad3/p/11094774.html