warmup

发表于 2021-01-04 分类于 Challenge， 2019， SCTF， Crypto
Challenge | 2019 | SCTF | Crypto | warmup

点击此处获得更好的阅读体验

题目考点

- *代码审计*
- *AES加密*

解题思路

打开审计一下源码，发现是 AES 加密。

```
#!/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.pad(message)))
    print 'so send your message:'
    msg = raw_input()
    print 'and your code:'
    code = raw_input()
    mac.identity(msg.decode('hex'), code)
    exit()
```

先给出了一段明文的 hex 和其加密之后的 hex，然后要求给出一段明文和其对应的密文，然后就就判断其为 `please send me your flag` 且密文正确的话就给出 flag。 这里很有意思的是这里

```
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')
```

对明文进行异或摘要到 16 位之后，才进行加密的。那么既然我们已知一组明文和密文，而且可以推算出其异或摘要之后获得的密钥，那么只要让我们传上去的明文摘要之后和前一组明文一致，那么就可以用前一组的密文来通过验证了。 对上面这个脚本进行改造，得到如下 POC 生成器

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-
from Crypto.Cipher import AES
from Crypto.Util.strxor import strxor
from Crypto.Random import get_random_bytes
flag = "fuck"
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
        # 最终目的 res 相等 24054d4c1a0f19444e0f4016080f1805
        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)
        print(res.encode('hex'))
        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.pad(message)))
    print 'so send your message:'
    msg = 'please send me your flag'
    print(msg)
    msg_o = msg + chr(63 - len(msg)) * (63 - len(msg))
    res = chr(0)*16
    for i in range(len(msg_o)/16 - 1):
        res = strxor(msg_o[i*16:(i+1)*16], res)
    msg_o = msg_o[:32] + strxor("24054d4c1a0f19444e0f4016080f1805".decode('hex'), res) + msg_o[48:]
    print(msg_o.encode('hex'))
    print 'and your code:'
    code = raw_input()
    mac.identity(msg.decode('hex'), code)
    exit()
```

解释一下，可以看到 code 那里我加了个 print ，输出第一组明文的十六位摘要和第二组明文的十六位摘要。而后对第二组明文进行二次摘要，对其加上一段十六位文本，让其异或之后与第一段明文的十六位摘要相等。再加上 1~15 个 pad，最后处理时利用 pad 保留下我们需要的文本 'please send me your flag'。

> *【前 32 位不用动】【32~48 位 拿来和前面异或，使得和前面已知密文的明文摘要一致】*
> *【48～（49～63） 位 拿来 padding，不是 64 位就是为了让这一段不参与前面的摘要计算，保证最后一位可控】*

运行，得到这段明文。

连接靶机，将明文和靶机返回的第一组明文提交，得到 flag。

- ***本文作者：*** *CTFHub*
- ***本文链接：*** *https://writeup.ctfhub.com/Challenge/2019/SCTF/Crypto/gvieEW8V7f7GJq3yJjhE6R.html*
- ***版权声明：*** *本博客所有文章除特别声明外，均采用 BY-NC-SA 许可协议。转载请注明出处！*

 # Challenge # 2019 # SCTF # Crypto
babygame
Maaaaaze