SECT CTF crypto Gsh思路分享

Lilac / 2018-09-14 22:04:22 / 浏览数 3687 技术文章 技术文章 顶(1) 踩(0)

SECT CTF crypto Gsh 思路分享

@(思路)[AES-ECB][hash forge][高斯消元]

> 题目描述：
> About last night...
> Service: nc 178.128.171.133 3333 | nc crypto.sect.ctf.rocks 3333
> Download: gsh.tar.gz
> Author: grocid

这道题是grocid出的，一直看着大佬的分享，在这谢谢大佬。
reddit.txt

```
Submitted 2 months ago by [deleted] to /r/infoleaks

I tried to login... was able to get a shell, but as a restricted user.
It seems horrendously badly configured. Which is what I would've expected.
Fortunately, I found a source code from an old unencrypted backup drive...
this one's particularly interesting...

   class AESHash(object):

       def __init__(self, key):
           self.bs = 16
           self.key = hashlib.sha256(key.encode()).digest()

       def _pkcs7pad(self, s, blksize=16):
           missing = abs(len(s) - (len(s) / blksize + 1) * blksize)
           return s + (chr(missing) * missing)

       def digest(self, user, password):
           cipher = AES.new(self.key, AES.MODE_ECB)
           q = 0
           data = self._pkcs7pad(user + password)
           for i in xrange(0, len(data), self.bs):
               block = data[i:i + self.bs]
               q ^= int(cipher.encrypt(block).encode("hex"), 0x10)
           return q

Their authentication mechanism uses some weird keyed AES-based MAC -- I've
never seen anything like it before. I'd say it's insecure, but I don't know
how to exploit it. Also, it's written in Python. Really?

Since the HMAC combines credentials in the following way... it's kind of
moot to give it a try. I've learnt from one-oh-one that h(message | key) is
secure... I think.  Motherf... I'll give up; it's late and I need to go to
sleep... over and out. For now.


-- JD
The revolution will not be televised.
```

信息搜集

> reddit上找了一通，没线索。那nc过去看看：

```
root@bin:/mnt/hgfs/CTF# nc 178.128.171.133 3333

Login (leave empty to create)
user:
Creating new user
user: 2
password:
```

```
IO error: cannot write 2:b5bbe950b1f52310ec0f986f4aeacbbb to /etc/shadow.
Logged in as 2
$ ls
-rw-r--r--    1 admin staff   27 Sep  5 19:31  flag.txt
-rw-------    1 admin staff   27 Sep  5 19:31  invoice.xls
$ cat flag.txt
-[--->+<]>--.[->+++++++<]>.--.>-[--->+<]>-.--[-->+++<]>.-------.------------.[-->+<]>---.[--->++<]>---.++[->+++<]>.+[-->+<]>+.
$ cat /etc/shadow
admin:8f643bbafa959617b12b591f3145e5c0
```

brainfuck 得到flag:SECT{th1s_1s_r34l_flag_1_Pr0mis3}.喵喵喵？提交是fake flag。

分析

```
class AESHash(object):

   def __init__(self, key):
       self.bs = 16
       self.key = hashlib.sha256(key.encode()).digest()

   def _pkcs7pad(self, s, blksize=16):
       missing = abs(len(s) - (len(s) / blksize + 1) * blksize)
       return s + (chr(missing) * missing)

   def digest(self, user, password):
       cipher = AES.new(self.key, AES.MODE_ECB)
       q = 0
       data = self._pkcs7pad(user + password)
       for i in xrange(0, len(data), self.bs):
           block = data[i:i + self.bs]
           q ^= int(cipher.encrypt(block).encode("hex"), 0x10)
       return q
```

题目提供了加密的oracle,可以任意_pkcs7pad(username+password)的AES-ECB hash,我们的目的是伪造出hash为8f643bbafa959617b12b591f3145e5c0的一串字符。
密码是弱密码？
要不用rockyou.txt爆破试试？
爆破了一段时间，主办方检测到了，，提示：不必用admin登陆
AES-ECB, 我们能得到任意对plaintext-ciphertext.然后恢复出16byte的Key，目前计算能力还不可能,参考
那接着想...
q ^= int(cipher.encrypt(block).encode("hex"), 0x10)
我们可以得到一定长度(>=128)的list,则我们需要的8f643bbafa959617b12b591f3145e5c0一定在这个list的sublist的xor里,子串的空间远大于目标大小。

```
In [15]: a = 0

In [16]: b = 200

In [17]: for i in range(1,200):
   ...:     a+=math.factorial(200)/(math.factorial(200-i)*math.factorial(i))
   ...:
   ...:

In [18]: print a
1606938044258990275541962092341162602522202993782792835301374

In [19]: a>n#n=0x8f643bbafa959617b12b591f3145e5c0
Out[19]: True

In [20]: a//n
Out[20]: 4722366482869645213695L
```

因为目标hash的长度为128bit,我们得到128对(plaintext-ciphertext)就能在ciphertext的子集里xor后得到目标hash.
这里想到了背包求解,构造着发现xor不能放入矩阵中，又不是超递增序列，xd,放弃。
换方向，要不是算法？
google看看
动态规划？
试了试，放弃了，空间换时间，但是需要的空间太大了。
bingo
128bit xor？高斯消元问题(我好菜啊)
转换成128个mod2的方程，生成方程脚本。

```
pubKey = [113602217935607289453201245405212005384L, 269903559201925776990330113614440744455L, 24156595894438163690164851676776
target = 0x8f643bbafa959617b12b591f3145e5c0
pubKey = pubKey[:128]
def _pkcs7pad(s, blksize=16):
   missing = abs(len(s) - (len(s) / blksize + 1) * blksize)
   return s + (chr(missing) * missing)
print len(pubKey)
for i in range(0,128):
   txt = ""
   for j in pubKey:
       if bin(j)[2:].zfill(128)[i] =="1":
           txt+="a"+str(pubKey.index(j))+"+"
   txt= txt[:-1]+"=="+bin(target)[2:].zfill(128)[i]
   print txt
```

这里我用wolfram mathematics求解(脚本见附件)

从结果可知，多解，C[1]为0，1均可，这里我取c[1]=0，得到了一组sublist :`out =`
`[2,4,5,9,10,11,12,13,16,17,22,24,27,28,30,31,32,35,36,38,40,43,45,47,48,49,51,52,54,57,59,62,63,64,68,71,73,75,79,80,81,87,89,`
验证结果：

```
In [71]: print a
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 35,

In [72]: a= [2,4,5,9,10,11,12,13,16,17,22,24,27,28,30,31,32,35,36,38,40,43,45,47
   ...: ,48,49,51,52,54,57,59,62,63,64,68,71,73,75,79,80,81,87,89,90,92,93,94,9
   ...: 7,98,99,101,103,105,106,111,115,116,117,118,120,121,122,123,124,126,127
   ...: ]

In [73]:

In [73]: txt=0
   ...: for i in a:
   ...:     txt^=pubKey[i]
   ...:

In [74]:

In [74]: print hex(txt)
0x8f643bbafa959617b12b591f3145e5c0L
```

求解:

```
from pwn import *

context.log_level = "debug"
# f = open("rockyou.txt","rb").read().split("\n")[::-1]
test = []
def _pkcs7pad(s, blksize):
   missing = abs(len(s) - (len(s) / blksize + 1) * blksize)
   return s + (chr(missing) * missing)
out = [2,4,5,9,10,11,12,13,16,17,22,24,27,28,30,31,32,35,36,38,40,43,45,47,48,49,51,52,54,57,59,62,63,64,68,71,73,75,79,80,81,
aeshash = ""
for i in out[:-1]:
   aeshash+=_pkcs7pad(str(i),16)
aeshash+=str(out[-1])
print len(aeshash)
print repr(aeshash)
for i in  range(130,140):
   io = remote("178.128.171.133", 3333)

   io.recvuntil("user: ")

   # io.sendline()

   # io.recvuntil("user: ")

   io.sendline(aeshash)
   io.recvuntil("password: ")

   io.sendline()

   data = io.recvuntil("Logged in as")

   io.interactive()
```

```
# $ cat invoice.xls
[DEBUG] Sent 0x10 bytes:
   'cat invoice.xls\n'
[DEBUG] Received 0x24 bytes:
   'SECT{...1_w4s_ly1ng_0f_c0urse_LuLz}\n'
SECT{...1_w4s_ly1ng_0f_c0urse_LuLz}
[DEBUG] Received 0x2 bytes:
   '# '
# $
```

总结: 算法需要加强，怼题不能放弃，不要以分数为目的而不停的换题做。