ALEXCTF-2017_cr2-many-time-secrets

原创

M3ng@L 于 2022-03-16 16:53:48 发布 1089 收藏

分类专栏： CTF比赛复现 文章标签： python Crypto

本文链接：https://blog.csdn.net/qq_51999772/article/details/123530427

ALEXCTF-2017_cr2-many-time-secrets

keywords: `many-time-pad`

Description

```
'''
0529242a631234122d2b36697f13272c207f2021283a6b0c7908
2f28202a302029142c653f3c7f2a2636273e3f2d653e25217908
322921780c3a235b3c2c3f207f372e21733a3a2b37263b313012
2f6c363b2b312b1e64651b6537222e37377f2020242b6b2c2d5d
283f652c2b31661426292b653a292c372a2f20212a316b283c09
29232178373c270f682c216532263b2d3632353c2c3c2a293504
613c37373531285b3c2a72273a67212a277f373a243c20203d5d
243a202a633d205b3c2d3765342236653a2c7423202f3f652a18
2239373d6f740a1e3c651f207f2c212a247f3d2e65262430791c
263e203d63232f0f20653f207f332065262c3168313722367918
2f2f372133202f1426652126372222220733e383f2426386b
'''
```

Analysis

根据题目意思，**是使用了相同的密钥对不同的明文进行加密**

本来一次一密`one time pad`是无懈可击的，但是这里使用了相同的密钥进行多次加密，是可以进行`many-time-pad-attack`的；`github`有现成脚本，或者使用`featherduster`解密软件直接进行该攻击得到`key`

以上是两种方法（但是`github`的脚本对于这道题有一些不足导致密钥和密文解出来不全，需要猜测）；

另外就是本题实际上已知了一部分`key`也就是`flag`前缀且本题的加密方式是**异或**

> ALEXCTF{

所以我们可以通过这部分的`key`来先解一解对应的密文，可以得到如下明文

```
'''
第0组cipher:b'Dear Fri'     第1组cipher:b'nderstoo' 第2组cipher:b'sed One ' 第3组cipher:b'n scheme'
第4组cipher:b'is the o' 第5组cipher:b'hod that' 第6组cipher:b' proven ' 第7组cipher:b'ever if '
第8组cipher:b'cure, Le' 第9组cipher:b'gree wit' 第10组cipher:b'ncryptio'
'''
```

很显然对于第0组来说，我们可以手动补全剩下的英文单词（标点符号是猜测的，可以通过之后的异或结果来判断是否在该位置上是正确的字符）

> Dear Friend,

那么把我们手动补全的英文单词与密文进行异或，那么得到的结果就是更多部分的`key`

```
known_part = "Dear Friend,"
second_part = xor(known_part,enc[0][:len(known_part)])
print(second_part)
'''
ALEXCTF{HERE
'''
```

所以以此类推，继续来这部分`key`来对11组密文进行异或（解密），把得到的部分明文结果以补全英文单词的方法手动扩展，逐渐把`key`全部恢复

The method of github

`github`上的`many-time-pad-attack`脚本：GitHub - Jwomers/many-time-pad-attack: Attacking A Many Time Pad - Cryptography

是`python2.x`环境下的，简单修改一下脚本

```
#!/usr/bin/env python
import string
import collections
import sets

# XORs two string
def strxor(a, b):     # xor two strings (trims the longer input)
    return "".join([chr(ord(x) ^ ord(y)) for (x, y) in zip(a, b)])

c1 = "0529242a631234122d2b36697f13272c207f2021283a6b0c7908"
c2 = "2f28202a302029142c653f3c7f2a2636273e3f2d653e25217908"
```

```
c3 = "322921780c3a235b3c2c3f207f372e21733a3a2b37263b313012"
c4 = "2f6c363b2b312b1e64651b6537222e37377f2020242b6b2c2d5d"
c5 = "283f652c2b31661426292b653a292c372a2f20212a316b283c09"
c6 = "29232178373c270f682c216532263b2d3632353c2c3c2a293504"
c7 = "613c37373531285b3c2a72273a67212a277f373a243c20203d5d"
c8 = "243a202a633d205b3c2d3765342236653a2c7423202f3f652a18"
c9 = "2239373d6f740a1e3c651f207f2c212a247f3d2e65262430791c"
c10 = "263e203d63232f0f20653f207f332065262c3168313722367918"
ciphers = [c1, c2, c3, c4, c5, c6, c7, c8, c9, c10]
# The target ciphertext we want to crack
# target_cipher = "263e203d63232f0f20653f207f332065262c3168313722367918"
target_cipher = ciphers
# target_cipher.append("2f2f372133202f1426652126372222220733e383f2426386")
# To store the final key
final_key = [None]*150
# To store the positions we know are broken
known_key_positions = set()

# For each ciphertext
for current_index, ciphertext in enumerate(ciphers):

 counter = collections.Counter()
 # for each other ciphertext
 for index, ciphertext2 in enumerate(ciphers):
  if current_index != index: # don't xor a ciphertext with itself
   for indexOfChar, char in enumerate(strxor(ciphertext.decode('hex'), ciphertext2.decode('hex'))): # Xor the tw
o ciphertexts
    # If a character in the xored result is a alphanumeric character, it means there was probably a space charac
ter in one of the plaintexts (we don't know which one)
    if char in string.printable and char.isalpha(): counter[indexOfChar] += 1 # Increment the counter at this in
dex
 knownSpaceIndexes = []

 # Loop through all positions where a space character was possible in the current_index cipher
 for ind, val in counter.items():
  # If a space was found at least 7 times at this index out of the 9 possible XORS, then the space character was
 likely from the current_index cipher!
  if val >= 7: knownSpaceIndexes.append(ind)
 #print knownSpaceIndexes # Shows all the positions where we now know the key!

 # Now Xor the current_index with spaces, and at the knownSpaceIndexes positions we get the key back!
 xor_with_spaces = strxor(ciphertext.decode('hex'),' '*150)
 for index in knownSpaceIndexes:
  # Store the key's value at the correct position
  final_key[index] = xor_with_spaces[index].encode('hex')
  # Record that we known the key at this position
  known_key_positions.add(index)

# Construct a hex key from the currently known key, adding in '00' hex chars where we do not know (to make a com
plete hex string)
final_key_hex = ''.join([val if val is not None else '00' for val in final_key])
# Xor the currently known key with the target cipher
print final_key_hex.decode('hex')
print final_key
for i in range(len(target_cipher)):
    output = strxor(target_cipher[i].decode('hex'),final_key_hex.decode('hex'))
# Print the output, printing a * if that character is not known yet
    print ''.join([char if index in known_key_positions else '*' for index, char in enumerate(output)])
```

得到结果

```
'''
ALEXTF{HRGESTHEKE}
Dear*Frie*d**T*is*tim* I*u
nder*tood*m**m*st*ke *nd*u
sed *ne t*m**p*d *ncr*pt*o
n sc*eme,*I**e*rd*tha* i*
is t*e on*y**n*ry*tio* m*t
hod *hat *s**a*he*ati*al*y
 pro*en t* ** *ot*cra*ke*
ever*if t*e**e* i* ke*t *e
cure* Let*M**k*ow*if *ou*a
gree*with*m**t* u*e t*is*e
'''
```

很显然得到的`flag`不是完整的，还需要进行手动补全（所以在这道题中这个的解题姿势不大好）

The method of featherduster

前面提到过`featherduster`是一个集成解密器，不支持`Windows`系统（不能直接通过`pip`下载）

环境配置`python2.x`，依赖`PyCrypto`，`ishell (which itself depends on readline and ncurses)`

下载地址：GitHub - nccgroup/featherduster: An automated, modular cryptanalysis tool; i.e., a Weapon of Math Destruction

指导：密码学分析和破解工具 FeatherDuster 使用说明 | 淡水网志 (restran.github.io)

`linux`命令行，下载好`featherduster`后，可以直接调用`featherduster`命令

```
featherduster xxx.txt # 导入数据
analyze # 分析密文能怎么攻破
use many-time-pad # 使用方法
run
results # 查看结果
```

~~我自己因为没有下载好`Python.h`所以没有实践，但确实是可以一步解出的；另外Crypto不建议使用工具来解CTF~~

The third method

推导以及思路在前面提到了，这里就直接放脚本了

```
# from Crypto.Util.strxor import strxor
from Crypto.Util.number import *

from pwn import *
f = open("C:\\Users\\Menglin\\Desktop\\f331d71a103f49bc94c2cc7838c29a9c","r")
enc = []
for line in f.readlines():
 enc.append(long_to_bytes(int(line.strip(),16)))
# print(enc)

# 这里写的函数是将步骤总结之后的集合，脚本实现的过程实际上就是在不断地重复该函数里面的内容
def find_part(known_flag,enc,known_msg=None):
 nums = 0
 for i in enc:
  print("第{}组cipher:".format(nums),end="")
  print(xor(known_flag,i[:len(known_flag)]),end="\t")
  nums += 1
 print('\n')
```

```
 if known_msg:
  next_part = xor(known_msg,enc[0][:len(known_msg)])
  print(next_part)
# find_part(first_part,enc)

first_part = "ALEXCTF{"
nums = 0
for i in enc:
 print("第{}组cipher:".format(nums),end="")
 print(xor(first_part,i[:len(first_part)]),end="\t")
 nums += 1
known_part = "Dear Friend,"
print('\n')

second_part = xor(known_part,enc[0][:len(known_part)])
print(second_part)
nums = 0
for i in enc:
 print("第{}组cipher:".format(nums),end="")
 print(xor(second_part,i[:len(second_part)]),end="\t")
 nums += 1
known_part = "sed One time pad"
print('\n')

third_part = xor(known_part,enc[2][:len(known_part)])
print(third_part)
nums = 0
for i in enc:
 print("第{}组cipher:".format(nums),end="")
 print(xor(third_part,i[:len(third_part)]),end="\t")
 nums += 1
print('\n')

known_part = "is the only encrypt"
forth_part = xor(known_part,enc[4][:len(known_part)])
print(forth_part)
nums = 0
for i in enc:
 print("第{}组cipher:".format(nums),end="")
 print(xor(forth_part,i[:len(forth_part)]),end="\t")
 nums += 1
print('\n')

known_part = "n scheme, I heard that"
fifth_part = xor(known_part,enc[3][:len(known_part)])
print(fifth_part)
nums = 0
for i in enc:
 print("第{}组cipher:".format(nums),end="")
 print(xor(fifth_part,i[:len(fifth_part)]),end="\t")
 nums += 1
print('\n')

known_part = "sed One time pad encryptio"
sixth_part = xor(known_part,enc[2][:len(known_part)])
print(sixth_part)
nums = 0
for i in enc:
 print("第{}组cipher:".format(nums),end="")
```

```
 print(xor(sixth_part,i[:len(sixth_part)]),end="\t")
 nums += 1
print('\n')
s
```

Reference

AlexCTF 2017 - CR2 Many time secrets (pequalsnp-team.github.io)

(11条消息) CTFlearn-ALEXCTF CR2: Many time secrets_fa1c4的博客-CSDN博客

Alex CTF 2017 Writeup: Many time secrets – 0xd13a – A rookie in a world of pwns