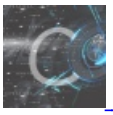
2021-07-12

原创

无名函数 于 2021-07-12 22:45:09 发布 148 收藏
分类专栏： Buu-crypto 文章标签： python

本文链接：https://blog.csdn.net/m0_57291352/article/details/118685025


Buu-crypto 专栏收录该内容
72 篇文章 1 订阅
订阅专栏

[ACTF新生赛2020]crypto-classic1

题目

hint

```
哇，这里有压缩包的密码哦，于是我低下了头，看向了我的双手，试图从中找到某些规律
xdfv ujko98 edft54 xdfv pok,.; wsdr43
```

解题

看向自己的双手，xdfv ujko98 edft54 xdfv pok,.; wsdr43

键盘加密，每组字母包围起来的字母分别为：circle

解压，得到：SRLU{LZPL_S_UASHKXUPD_NXYTFTJT}

提示是维吉尼亚加密，

```
c='SRLU{LZPL_S_UASHKXUPD_NXYTFTJT}'
m='ACTF{'
a=[]
for i in range(4):
    a.append(str(ord(c[i])-ord(m[i])))
print(m,end='')
for i in range(5,len(c)):
    if 'A'<= c[i]<= 'Z':
        print(chr((ord(c[i])-int(a[i%4])-ord('A'))%26+ord('A')),end='')
    else:
        print(c[i],end='')
```

得到：ACTF{WHAT_A_CLASSICAL_VIGENERE}
最后将其中字母换成小写就行了

答案

flag{what_a_classical_vigenere}

[AFCTF2018]你听过一次一密么？

题目

Problem

```
25030206463d3d393131555f7f1d061d4052111a19544e2e5d
0f020606150f203f307f5c0a7f24070747130e16545000035d
1203075429152a7020365c167f390f1013170b1006481e1314
0f4610170e1e2235787f7853372c0f065752111b15454e0e09
081543000e1e6f3f3a3348533a270d064a02111a1b5f4e0a18
0909075412132e247436425332281a1c561f04071d520f0b11
4116111b101e2170203011113a69001b475206011552050219
0410060646122970203754533 42c17545a01451811411a470e
021311114a5b0335207f7c167f22001b44520c15544801125d
06140611460c26243c7f5c167f3d0154460100530059071 45d
0f05110d160f263f3a7f4210372c03111313090415481d49
```

解题

一次一密是无条件安全的。

那这道题应该是二次密码本加密了

参考关于Many-Time-Pad的某大佬笔记、关于ACTF解题笔记

写脚本中、、、、

算了，我放弃，还是来看大佬的吧：

```
import string
import collections
import sets, sys

# 11 unknown ciphertexts (in hex format), all encrpyted with the same key

c1='25030206463d3d393131555f7f1d061d4052111a19544e2e5d'
c2='0f020606150f203f307f5c0a7f24070747130e16545000035d'
c3='1203075429152a7020365c167f390f1013170b1006481e1314'
c4='0f4610170e1e2235787f7853372c0f065752111b15454e0e09'
c5='081543000e1e6f3f3a3348533a270d064a02111a1b5f4e0a18'
c6='0909075412132e247436425332281a1c561f04071d520f0b11'
c7='4116111b101e2170203011113a69001b475206011552050219'
c8='0410060646122970203754533 42c17545a01451811411a470e'
c9='021311114a5b0335207f7c167f22001b44520c15544801125d'
c10='06140611460c26243c7f5c167f3d0154460100530059071 45d'
c11='0f05110d160f263f3a7f4210372c03111313090415481d49'
ciphers = [c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11]
# The target ciphertext we want to crack
#target_cipher = "0529242a631234122d2b36697f13272c207f2021283a6b0c7908"

# 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)])

def target_fix(target_cipher):
 # 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 t
wo ciphertexts
     # If a character in the xored result is a alphanumeric character, it means there was probably a space chara
cter 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 i
ndex
  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 wa
s 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 co
mplete 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
 output = strxor(target_cipher.decode('hex'),final_key_hex.decode('hex'))

 print ("Fix this sentence:")
 print (''.join([char if index in known_key_positions else '*' for index, char in enumerate(output)])+"n")

 # WAIT.. MANUAL STEP HERE
 # This output are printing a * if that character is not known yet
 # fix the missing characters like this: "Let*M**k*ow if *o{*a" = "cure, Let Me know if you a"
 # if is too hard, change the target_cipher to another one and try again
 # and we have our key to fix the entire text!

 #sys.exit(0) #comment and continue if u got a good key

 target_plaintext = "cure, Let Me know if you a"
 print ("Fixed:")
 print (target_plaintext+"n")

 key = strxor(target_cipher.decode('hex'),target_plaintext)

 print ("Decrypted msg:")
 for cipher in ciphers:
  print (strxor(cipher.decode('hex'),key))

 print ("nPrivate key recovered: "+key+"n")

for i in ciphers:
 target_fix(i)
```

运行得到：
afctf{OPT_1s_Int3rest1ng}

答案

flag{OPT_1s_Int3rest1ng}