* **解题文档**
* **关键技术**

解题主要掌握pyo文件反编译，加密代码混淆还原，随机算法安全性分析以及密钥蛮力攻击，难点是分析随机算法安全性并编写密钥蛮力攻击脚本。

* **解题思路**

1. 还原加密算法
2. 分析加密算法
3. 编写攻击脚本

* **解题步骤**

1. 还原加密算法



# uncompyle6 version 2.11.2  
# Python bytecode 2.7 (62211)  
# Decompiled from: Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:53:40) [MSC v.1500 64 bit (AMD64)]  
# Embedded file name: encrypt.py  
# Compiled at: 2017-07-11 17:19:27  
*from* random *import* randint  
*from* math *import* floor, sqrt  
\_ = ''  
\_\_ = '\_'  
\_\_\_\_ = [ ord(\_\_\_) *for* \_\_\_ *in* \_\_ ]  
\_\_\_\_\_ = randint(65, max(\_\_\_\_)) \* 255  
*for* \_\_\_ *in* range(len(\_\_)):  
 \_ += str(int(floor(float(\_\_\_\_\_ + \_\_\_\_[\_\_\_]) / 2 + sqrt(\_\_\_\_\_ \* \_\_\_\_[\_\_\_])) % 255)) + ' '  
  
*print* \_  
# okay decompiling encrypt.pyo

代码中做了混淆导致可读性很低，尝试进行代码还原：

*from* random *import* randint  
*from* math *import* floor, sqrt  
  
ciphertext = ''  
plaintext = '\_'  
array = [ord(i) *for* i *in* plaintext]  
key = randint(65, max(array)) \* 255  
*for* i *in* range(len(plaintext)):  
 ciphertext += str(int(floor(float(key + array[i]) / 2 + sqrt(key \* array[i])) % 255)) + ' '  
*print* ciphertext

1. 分析加密算法

通读分析以上代码可知，核心加密算法的代码：

key = randint(65, max(array)) \* 255  
*for* i *in* range(len(plaintext)):  
 ciphertext += str(int(floor(float(key + array[i]) / 2 + sqrt(key \* array[i])) % 255)) + ' '

密钥key虽然是生成随机的，但是并不安全，因为它的生成范围为(65, max(array))，而array为可显明文的十进制ASCII值，也就是说key的最大生成范围为(65, 127)，而以上代码是逐个字符进行加密，且每次都是使用同一个密钥key，在已知密文的情况下我们可以考虑使用蛮力攻击爆破明文。计算次数为：

19 (明文长度) \* (127 - 65)(可能明文) \* (127 - 65)(可能密钥) = 70360



* **结果提交**

flag{ThisRandomIsNotSafe}