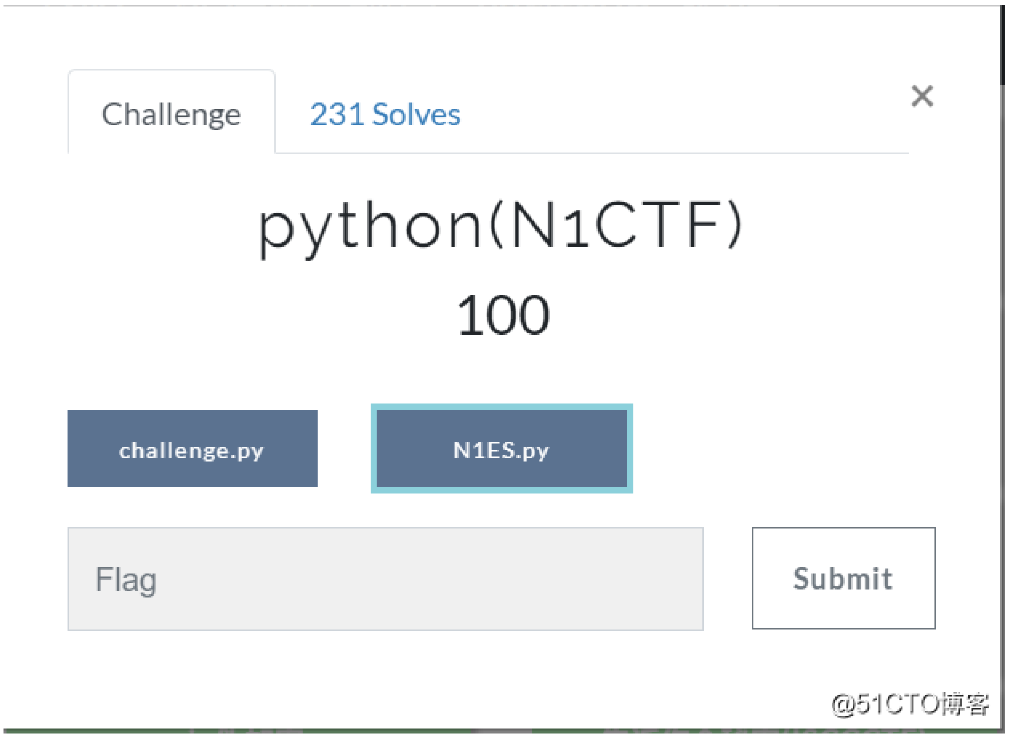
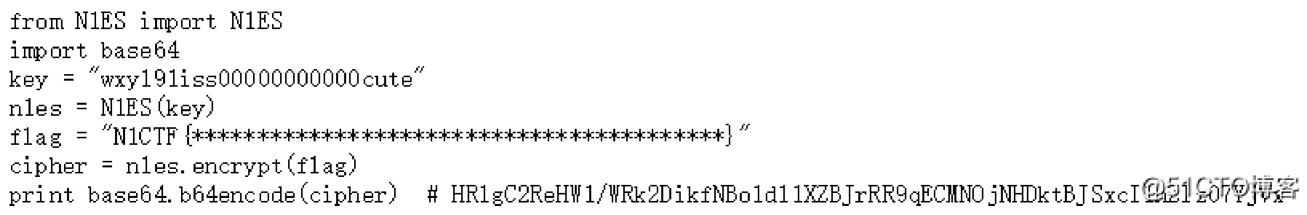
Bugku 加密 python writeup

转载

[baoju9513](http://blog.51cto.com/13992485/2434946) 于 2019-09-03 10:27:13 发布 113 收藏
文章标签： python
原文链接：http://blog.51cto.com/13992485/2434946
版权

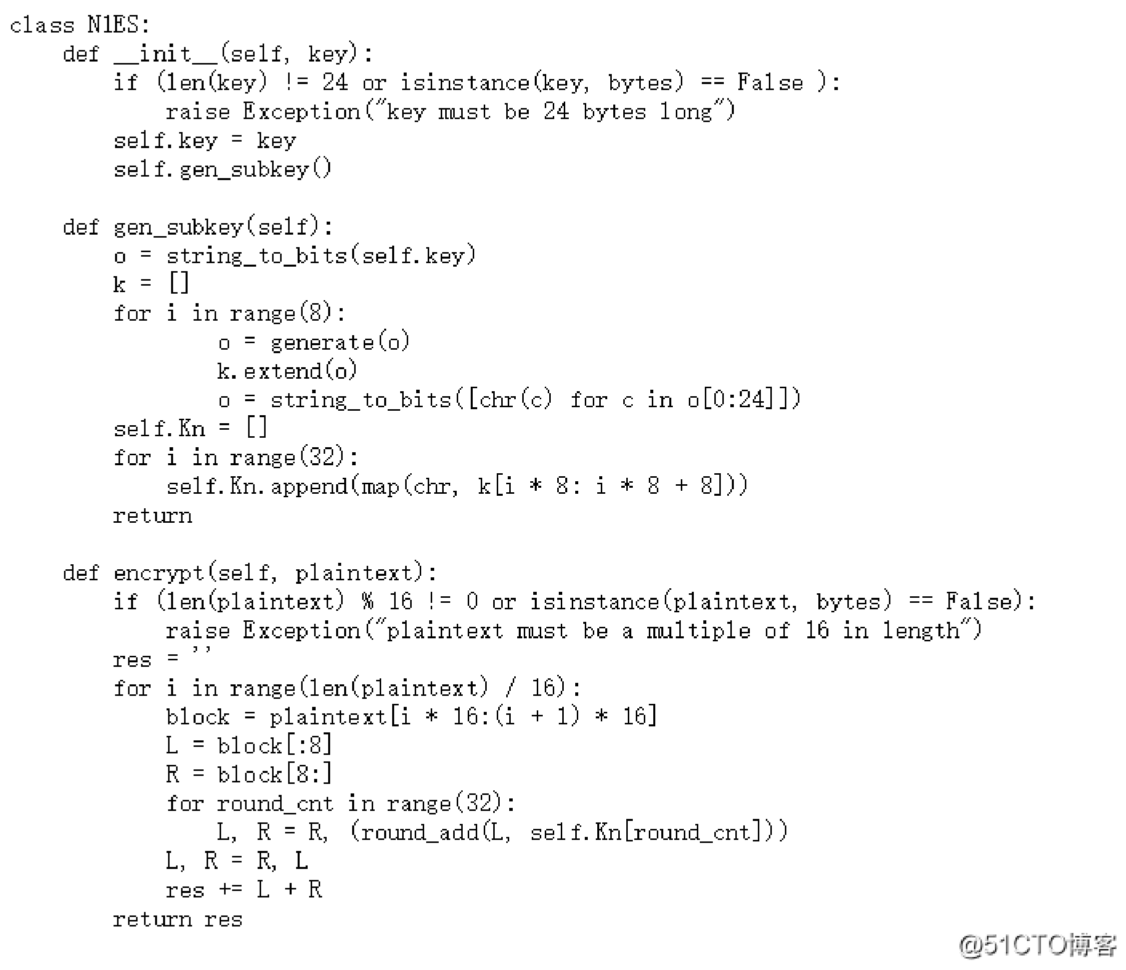


一上来就给了两个文件，一个是加密的源代码，一个是加密过程文件，



```
from N1ES import N1ES
import base64
key = "wxy191iss00000000000cute"
n1es = N1ES(key)
flag = "N1CTF{*******************************************}"
cipher = n1es.encrypt(flag)
print base64.b64encode(cipher)  # HR1gC2ReHW1/WRk2DikfNBo1d11XZBJrRR9qECMNOjNHDktBJSxcI1hZIz07YjVx
```

challenge.py



```python
class N1ES:
    def __init__(self, key):
        if (len(key) != 24 or isinstance(key, bytes) == False ):
            raise Exception("key must be 24 bytes long")
        self.key = key
        self.gen_subkey()

    def gen_subkey(self):
        o = string_to_bits(self.key)
        k = []
        for i in range(8):
                o = generate(o)
                k.extend(o)
                o = string_to_bits([chr(c) for c in o[0:24]])
        self.Kn = []
        for i in range(32):
            self.Kn.append(map(chr, k[i * 8: i * 8 + 8]))
        return

    def encrypt(self, plaintext):
        if (len(plaintext) % 16 != 0 or isinstance(plaintext, bytes) == False):
            raise Exception("plaintext must be a multiple of 16 in length")
        res = ''
        for i in range(len(plaintext) / 16):
            block = plaintext[i * 16:(i + 1) * 16]
            L = block[:8]
            R = block[8:]
            for round_cnt in range(32):
                L, R = R, (round_add(L, self.Kn[round_cnt]))
            L, R = R, L
            res += L + R
        return res
```

N1ES.py

N1ES.py里一共有四个函数，一个类，类里含有两个函数，除了最后一个encrypt函数外其他函数都是在对key进行运算，然后通过key来对flag进行加密，所以我直接跑了一下程序，获得了key加密后的数据，然后只对encrypt函数进行逆向
解密脚本：

```
Kn=[['~', 'w', 'Y', 'k', 'k', '\x02', '\x05', '\x05'],['w', 'd', '}', '\x14', '?', '\x13', '\x04', 'W'],['l

import base64

s=base64.b64decode('HRlgC2ReHW1/WRk2DikfNBo1dl1XZBJrRR9qECMNOjNHDktBJSxcI1hZIz07YjVx')

flag=[]

for i in range(3):

    flag.append(s[i*16:(i+1)*16])

from z3 import *

def fun(a,b):

    x=[BitVec('x%d'%i,32) for i in range(8)]

    solver=Solver()

    res=''

    for i in range(len(a)):

        exec("solver.add(x[i]-2*(x[i]&ord(b[i]))==ord(a[i])-ord(b[i]))")

        solver.check()

        try:

            exec("res+=chr(solver.model()[x[i]].as_long())")

        except:

            print solver

    return res

res=''

for i in flag:

    L=i[:8]

    R=i[8:]

    L,R=R,L

    for k in range(32):

        L,R=R,fun(L,Kn[k])

    res+=L+R

print res
```