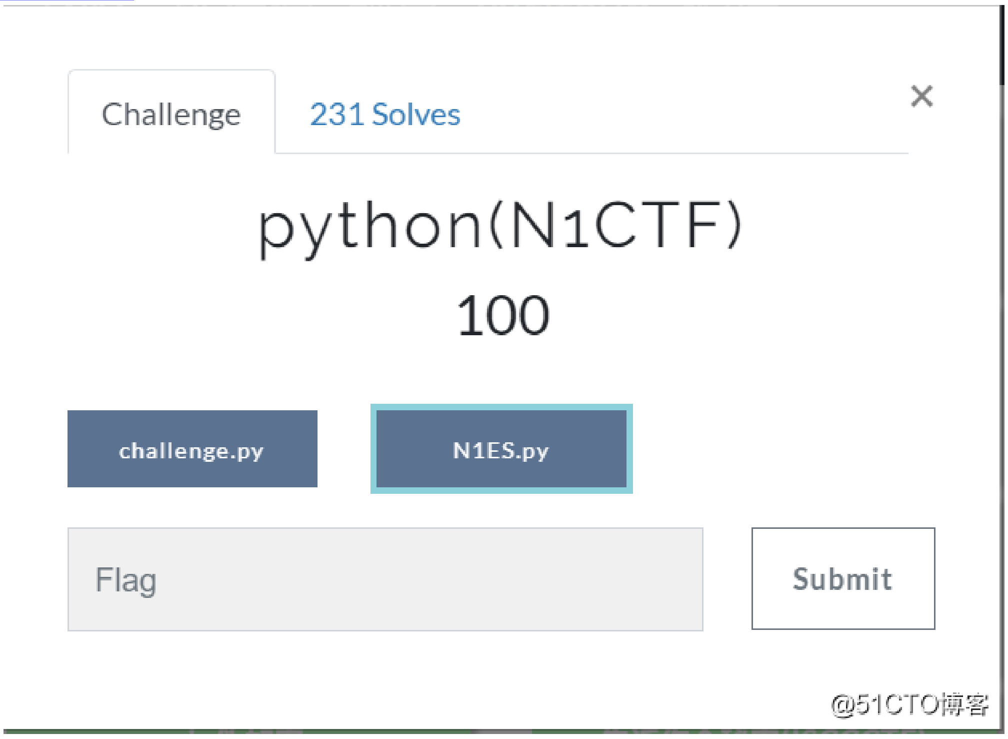
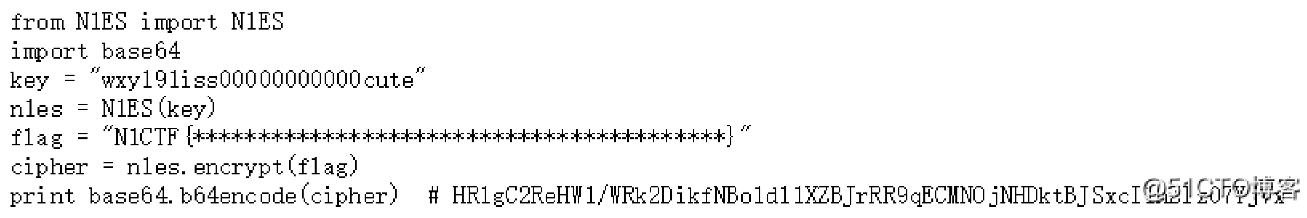
python 开发板加密_Bugku 加密 python writeup

weixin_39956558 于 2020-11-30 11:40:41 发布 20 收藏
文章标签： python 开发板加密

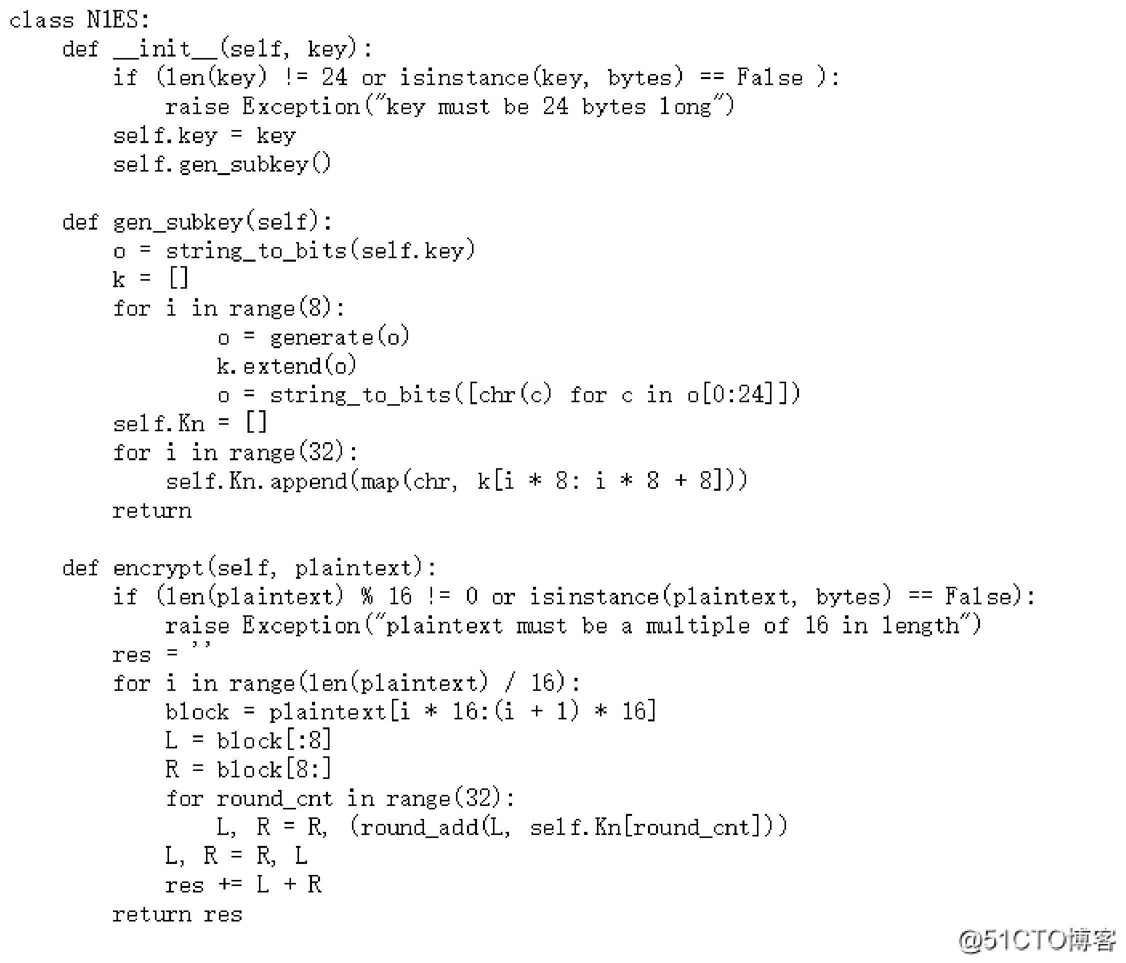


一上来就给了两个文件，一个是加密的源代码，一个是加密过程文件，



```
from N1ES import N1ES
import base64
key = "wxy191iss00000000000cute"
n1es = N1ES(key)
flag = "N1CTF{******************************************}"
cipher = n1es.encrypt(flag)
print base64.b64encode(cipher)  # HR1gC2ReHW1/WRk2DikfNBo1dl1XZBJrRR9qECMNOjNHDktBJSxcI1hZIz07YjVx
```

challenge.py



```
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', '6', '\x08', '\x04', '\x13', '3', '\x19', '\x10'],['\x08', 'P', '2', '\x02', '/', 'W', '/', 'W'],['\x08', '\x14', '?', '@', 'W', '^', ' ', 'k'],['\x1b', '6', '^', '(', 'M', 'Y', '\x19', '\x02'],['3', 'f', 'w', '(', '\x13', '}', '\x08', 'u'],['=', '_', '\x13', 'M', '2', '=', '@', '\x04'],['z', '_', '~', '\x08', 'L', 'f', '\x19', 'z'],['l', 'Y', '\x01', '}', '/', '}', 'L', 'o'],['\x19', '\x05', '3', '\x01', 'z', 'w', '~', '?'],['L', 'B', '~', '\x13', '@', '6', '@', '\x05'],['\x08', 'd', '\x13', 'L', '^', '?', 'L', 'u'],['\x05', '{', 'M', 'P', 'M', '\n', 'z', 'P'],['k', '~', 'k', '/', 'o', 'u', '\x19', '\x04'],['o', 'k', '(', '\x13', 'l', 'f', ' ', '='],['~', '\x04', '\x08', '^', '\x02', '\n', '6', '3'],['/', '\x05', 'w', '2', ' ', 'd', '\x13', '6'],[' ', '/', '}', '?', '\x04', '}', 'z', '\x19'],['\x05', '\n', '\n', 'l', '\x02', 'l', '^', 'l'],['k', '3', '}', '\x19', 'u', 'l', ' ', '^'],['~', 'B', '\x02', '}', 'k', '\x05', '\x02', '/'],['\n', '\x05', '^', '^', 'P', '}', '!', '{'],['\x08', 'W', 'u', 'o', ' ', '2', 'd', '\x04'],['/', 'W', 'w', '\x08', 'z', '\x19', '@', 'l'],['\x14', ' ', 'P', '!', '6', '6', ' ', '}'],['(', '!', '\x01', '\x08', 'd', '\x08', 'w', '?'],['u', 'W', '@', '\x13', '}', '~', '6', 'o'],['3', 'B', 'd', '\x01', 'W', '2', '\n', '6'],['}', '\x08', '6', '\x19', '&', '\x04', 'k', 'u'],['\x13', '2', '2', '(', '\x19', '{', '/', 'w'],['\x02', 'Y', ' ', 'W', '\x08', 'u', '\x01', '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
```