2019全国大学生信息安全竞赛初赛writeup

原创

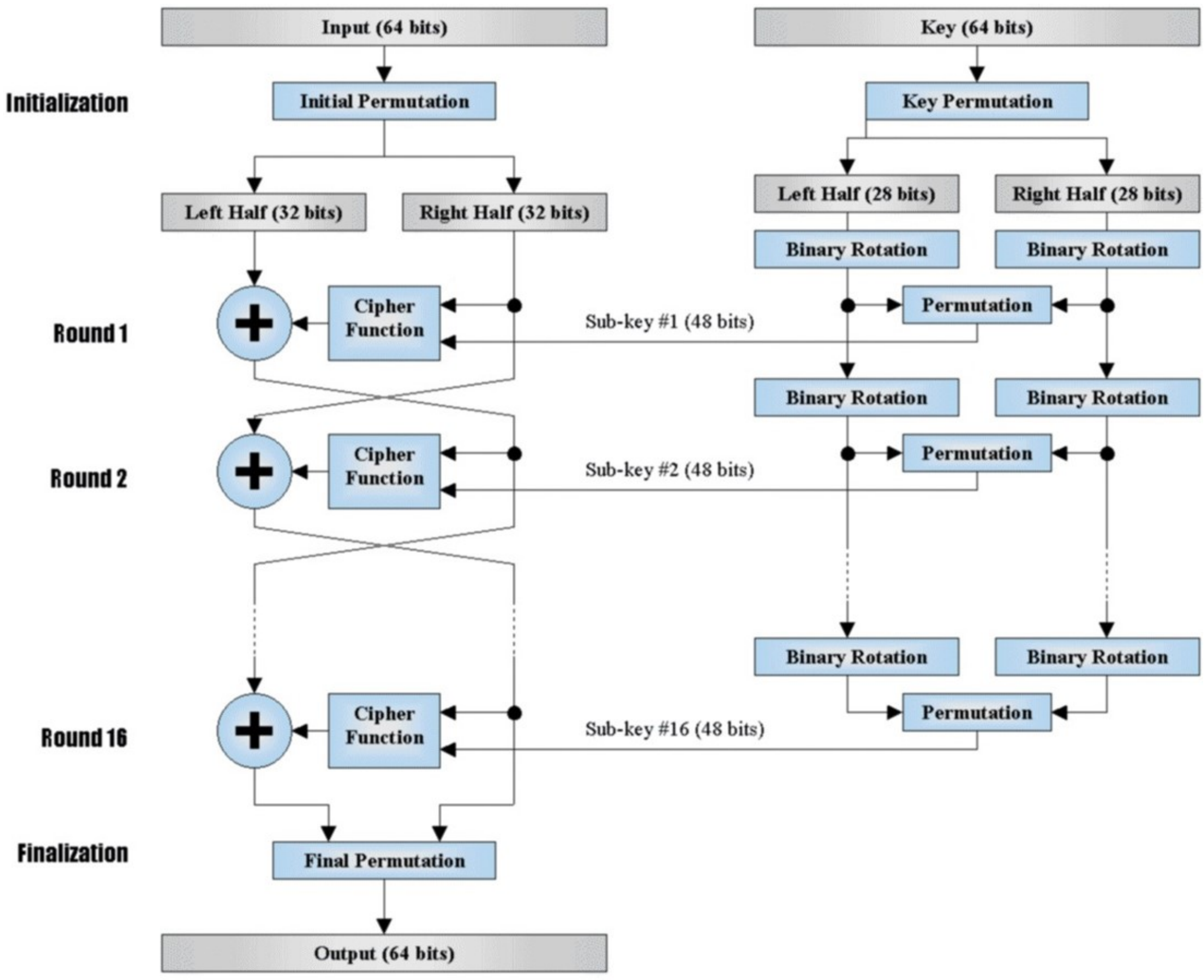
a16511232 于 2019-05-05 11:15:00 发布 3373 收藏 4

part_des

- 题目描述

```
Round n part_encode-> 0x92d915250119e12b
Key map -> 0xe0be661032d5f0b676f82095e4d67623628fe6d376363183aed373a60167af537b46abc2af53d97485591f5bd94b94
```

按题意keymap就是des加密过程中生成的子密钥Kn
n应该是des加密过程16轮循环内的某一个临时值



des加密过程

有一个pyDes库可以进行常规的des加解密，我们在此基础上修改比较容易
先对n和keymap进行处理，转换为二进制

```
print(bin(n)[2:])
print(bin(keymap)[2:])
```

然后看到一篇文章，可以将子密钥转化为原始密钥
由子密钥反推deskey

```
代码如下：
import libnum
import binascii

key1 = [1,1,1,0,0,0,0,0,1,0,1,1,1,1,1,0,0,1,1,0,0,1,1,0,0,0,0,1,0,0,0,0,0,0,1,1,0,0,1,0,1,1,0,1,0,1,0,1,]
__pc2 = [
        13, 16, 10, 23,  0,  4,
         2, 27, 14,  5, 20,  9,
        22, 18, 11,  3, 25,  7,
        15,  6, 26, 19, 12,  1,
        40, 51, 30, 36, 46, 54,
        29, 39, 50, 44, 32, 47,
        43, 48, 38, 55, 33, 52,
        45, 41, 49, 35, 28, 31
    ]
C1D1 = ['*']*56
for i in range(0,len(key1)):
    C1D1[__pc2[i]] = key1[i]
print C1D1

C0='000000001*11111111*111*10*00'
D0='0000111*11*1001*0000100001*0'

__pc1 = [56, 48, 40, 32, 24, 16, 8,
         0, 57, 49, 41, 33, 25, 17,
         9, 1, 58, 50, 42, 34, 26,
         18, 10, 2, 59, 51, 43, 35,
         62, 54, 46, 38, 30, 22, 14,
         6, 61, 53, 45, 37, 29, 21,
         13, 5, 60, 52, 44, 36, 28,
         20, 12, 4, 27, 19, 11, 3
         ]
C0D0 = C0+D0
res = ['*']*64
deskey = ""
for i in range(0,len(__pc1)):
    res[__pc1[i]] = C0D0[i]
for i in res:
    deskey += i
print deskey

def zuoyiwei(str,num):
    my = str[num:len(str)]
    my = my+str[0:num]
    return my

def key_change_1(str):
    key1_list = [57,49,41,33,25,17,9,1,58,50,42,34,26,18,10,2,59,51,43,35,27,19,11,3,60,52,44,36,63,55,47,3
    res = ""
    for i in key1_list:
        res+=str[i-1]
    return res

def key_change_2(str):
    key2_list = [14,17,11,24,1,5,3,28,15,6,21,10,23,19,12,4,26,8,16,7,27,20,13,2,41,52,31,37,47,55,30,40,51
    res = ""
    for i in key2_list:
        res+=str[i-1]
    return res
def key_gen(str):
    key_list = []
    key_change_res = key_change_1(str)
    key_c = key_change_res[0:28]
    key_d = key_change_res[28:]
    for i in range(1,17):
        if (i==1) or (i==2) or (i==9) or (i==16):
            key_c = zuoyiwei(key_c,1)
            key_d = zuoyiwei(key_d,1)
        else:
            key_c = zuoyiwei(key_c,2)
            key_d = zuoyiwei(key_d,2)
        key_yiwei = key_c+key_d
        key_res = key_change_2(key_yiwei)
        key_list.append(key_res)
    return key_list
#01100***01**011*0111001*0110101*0110010*01*00*0*0*1*010*0110010*

deskey = '01100abc01de011f0111001g0110101h0110010i01j00k0l0m1n010o0110010p'
print key_gen(deskey)

deskey = '0110011c0110011f0111001g0110101h0110010i0110010L0111010o0110010p'

def bintostr(str):
    res = ""
    for i in range(0,len(str),8):
        res += chr(int(str[i:i+8],2))
    return res
for c in "01":
    for f in "01":
        for g in "01":
            for h in "01":
                for i in "01":
                    for L in "01":
                        for o in "01":
                            for p in "01":
                                str = '0110011'+c+'0110011'+f+'0111001'+g+'0110101'+h+'0110010'+i+'0110010'
                                str = bintostr(str)
                                print str
```

输出如下：

```
ffrjddtd
ffrjddte
ffrjddud
ffrjddue
ffrjdetd
ffrjdete
ffrjdeud
...
```

跟文章不一样，密钥全是可以显示的字符
将字符再转换成Kn，所有Kn都是一样的，有些奇怪
后来搜索了一下，发现密钥的第8位，16位，24位好像是不影响加密的，所以这么多个密钥都可以用来加解密，结果完全一样

接下来把pyDes.py的全部代码复制进来，再加上如下代码
这里是一个标准的des加解密流程，先随便加密8字节的明文12345678

```
key='ffskeeue'
result = des(key)
y = result.encrypt('12345678')
print(y)
z = result.decrypt(y)
print(z)
```

然后修改des类中的__des_crypt函数，也就是加密实际执行的函数，插入题目所给的n

```
        i = 0
        while i < 16:
            ############
            if(i==13 and crypt_type == des.ENCRYPT):
                            self.L=[1,0,0,1,0,0,1,0,1,1,0,1,1,0,0,1,0,0,0,1,0,1,0,1,0,0,1,0,0,1,0,1]
                            self.R=[0,0,0,0,0,0,0,1,0,0,0,1,1,0,0,1,1,1,1,0,0,0,0,1,0,0,1,0,1,0,1,1]
                            print(i,self.L,self.R)
            ############
            tempR = self.R[:]
```

这里的i==13是从1开始一个一个试出来的
在等于13时，解密结果为`y0ur9Ood`，看起来很像flag
包上`flag{y0ur9Ood}`，交上去就对了

其实Kn反解出密钥应该不是必要步骤，在pyDes里面覆盖掉Kn应该也可以
测试确实可以

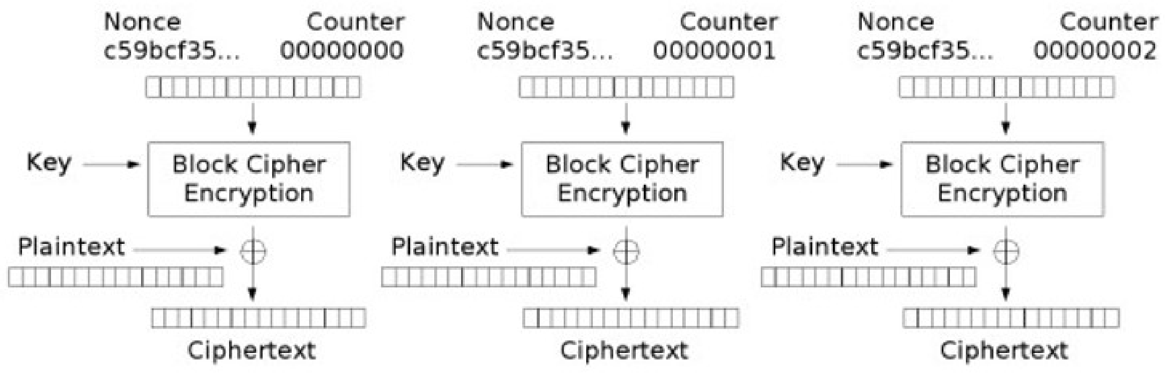
---

warmup

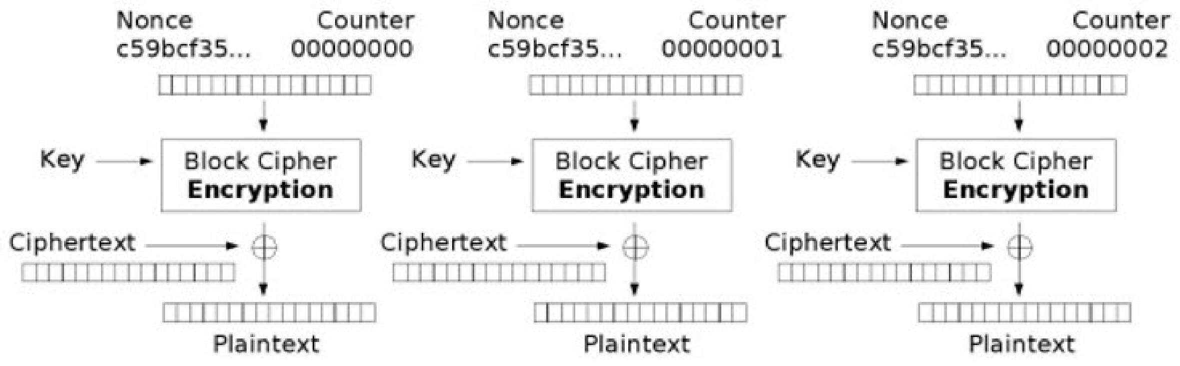
- 题目描述

```
from Crypto.Cipher import AES
from Crypto.Util import Counter
from Crypto import Random
import binascii
import SocketServer

pad = lambda s: s + (16 - len(s) % 16) * chr(16 - len(s) % 16)
flag = "*******************************************"
key = Random.get_random_bytes(16)
print binascii.b2a_hex(key)
prefix = Random.get_random_bytes(4)
suffix = Random.get_random_bytes(4)

def enc(plaintext):
    count = Counter.new(64, prefix=prefix, suffix=suffix)
    cipher = AES.new(key, AES.MODE_CTR, counter=count)
    print(binascii.hexlify(pad(plaintext)))
    return cipher.encrypt(pad(plaintext + flag))

class ThreadedTCPServer(SocketServer.ThreadingMixIn, SocketServer.TCPServer):
    pass

class EncHandler(SocketServer.BaseRequestHandler):
    def handle(self):
        self.request.sendall("Welcome to flag getting system\n")
        while 1:
            self.request.sendall("plaintext>")
            plaintext = self.request.recv(1024).strip()
            ciphertext = binascii.hexlify(enc(plaintext))
            self.request.sendall("result>" + ciphertext + '\n')

if __name__ == "__main__":
    HOST, PORT = "0.0.0.0", 7777
    server = ThreadedTCPServer((HOST, PORT), EncHandler)
    server.serve_forever()
```

就是aes加密，输入明文，服务器返回明文+flag加密后的密文
加密模式是CTR，8字节一组



Counter (CTR) mode encryption



Counter (CTR) mode decryption

AES-CTR
每次连接后，密钥流是不变的，如果我们一开始先输入空明文
那么加密的就是`AES(''+flag)`，记录为`str1`
然后再输入比flag长度长的`fffffffffffffffffffffffffffffffffff`
加密的就是`AES("fffffffffffffffffffffffffffffffffff"+flag)`，记录为`str2`

将`str2`与`fffffffffffffffffffffffffffffffffff`异或，就得到密钥流
`keystream = xor(str2, "fffffffffffffffffffffffffffffffffff")`
再将密钥流与`str1`异或，就得到flag
`flag = xor(keystream, str1)`

题目不难，但是对bytes型，str型，二进制，各种类型转换不熟练，卡了好一会
脚本如下：

```
#python3
import binascii
import itertools

ffff = 'fffffffffffffffffffffffffffffffffffffffffffffffff'
print(ffff)

miwenf = binascii.unhexlify('0e3ecd7afef871...')
print(miwenf)

miwen = binascii.unhexlify('0e34ca7be3a7272b37da7245bbbf9.......')
print(miwen)

def xor(s, key):
    key = key * (len(s) / len(key) + 1)
    return ''.join(chr(ord(x) ^ ord(y)) for (x,y) in itertools.izip(s, key))

keyliu = xor(miwenf,ffff)
print(keyliu)

flag = xor(miwen,keyliu)
print(flag)
```

Asymmetric

- 题目描述

```
import gmpy2
import random
from Crypto.Util.number import *
from flag import flag

def generate_key(nbit):
    p = getPrime(nbit)
    r = random.randint(2, 10)
    s = random.randint(r, nbit)
    while True:
        e = random.randint(3, p**r*(p-1))
        if gmpy2.gcd(e, p**s*(p-1)) == 1:
            break
    pubkey = (long(e), long(p**r))
    return pubkey

def crypt(msg, pkey):
    e, n = pkey
    m = bytes_to_long(msg)
    assert m < n - 1
    enc = pow(m, e, n)
    return long_to_bytes(enc)

nbit = 1024
pubkey = generate_key(nbit)
print 'pubkey =', pubkey
msg = flag
enc = crypt(msg, pubkey)
print 'enc =\n', enc.encode('base64')
```

```
pubkey = (5813456741606134624642495055280695995216414187398819703833931817237351409625882330046879172605137
enc = YXmuOsaD1W4poL...
```

刚开始给的题目里没有flag，随后更新了题目。
一开始以为只是添加了flag，做了半天发现解不出来，才发现p和e都更新了，以后做题细心点。
这题的p在factordb.com可解`(1657407551...67<309>)^4`，则`r=4`

有点类似RSA，猜测欧拉函数就是`p**s*(p-1)`，则`d = gmpy2.invert(e,p**s*(p-1))`，测试可解，代码如下：

```
import gmpy2
import random
from Crypto.Util.number import *
import binascii

p=165740755190793304655854506052794072378181046252118367693457385632818329041540419488625472007710062128632
r=4
e=581345674160613462464249505528069599521641418739881970383393181723735140962588233004687917260513782647159

n=p**r

for s in xrange(r,1024):
    if gmpy2.gcd(e, p**s*(p-1)) == 1:
        break

d = gmpy2.invert(e,p**s*(p-1))

enc ='YXmuOsaD1...'
enc = enc.decode('base64')

def decrypt(msg, d):
    m = bytes_to_long(msg)
    #assert m < n - 1
    dec = pow(m, d, n)
    return long_to_bytes(dec)

flag = decrypt(enc,d)
print flag
```

deep_leaning

- 题目描述

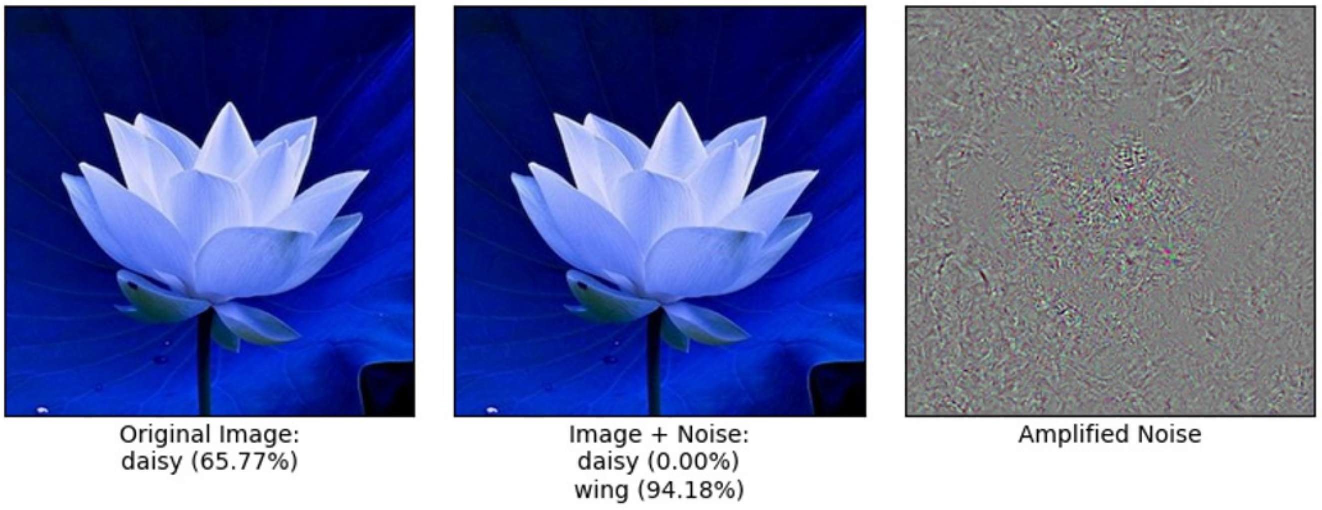
```
from PIL import Image
import sys
import os
import numpy as np
import random
import time
import base64
import inception
import string
import hashlib
SALT_LEN = 10
HEX_LEN = 4

std_image_name="/home/ctf/images/image.jpg"
input_image_name=''
os.environ['TF_CPP_MIN_LOG_LEVEL']='2'

def base_str():
    return "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"

def random_string(length):
    string = [random.choice(base_str()) for i in range(length)]
    return ("".join(string))
```

```python

def tofile(data):
    try:
        data=base64.b64decode(data)
    except:
        return ""
    filename="/backup/"+str(time.time())+".jpg"
    fd=open(filename,"wb")
    fd.write(data)
    fd.close()
    filename="/home/ctf/images/"+str(time.time())+".jpg"
    fd=open(filename,"wb")
    fd.write(data)
    fd.close()
    return filename

def check():
    global input_image_name
    try:
        input_image=Image.open(input_image_name)
        std_image=Image.open(std_image_name)
    except:
        print("[-]give me a real image!!")
        sys.stdout.flush()
        return False
    input_image_np=np.array(input_image)
    std_image_np=np.array(std_image)
    input_x=len(input_image_np)
    input_y=len(input_image_np[0])
    input_z=len(input_image_np[0][0])
    std_x=len(std_image_np)
    std_y=len(std_image_np[0])
    std_z=len(std_image_np[0][0])
    if std_x!=input_x or std_y!=input_y or std_z!=input_z:
        return False
    diff=0
    for i in range(input_x):
        for j in range(input_y):
            for k in range(input_z):
                if input_image_np[i][j][k]>std_image_np[i][j][k]:
                    diff+=input_image_np[i][j][k]-std_image_np[i][j][k]
                else:
                    diff+=std_image_np[i][j][k]-input_image_np[i][j][k]
    diff=diff/(input_x*input_y*input_z)
    if diff>0.8:
        return False
    return True

def classify(image_path):
    model = inception.Inception()
    pred = model.classify(image_path=image_path)
    return (pred.argmax(),model.name_lookup.cls_to_name(pred.argmax(),only_first_name=True))
def getflag():
    fd=open("/home/ctf/flag")
    flag=fd.readline()
    fd.close()
    return flag
def main():
    salt=random_string(SALT_LEN)
    tmpvalue=random_string(20)+salt
    md5=hashlib.md5()
    md5.update(tmpvalue.encode("utf-8"))
    submd5=md5.hexdigest()[:4]
    print ("[*]Proof of work:")
    print ("\tMD5(key+\"%s\")[:4]==%s"%(salt,submd5))
    print ("[+]Give me the key:")
    sys.stdout.flush()
    value=sys.stdin.readline()[:-1]
    value=value+salt
    md5=hashlib.md5()
    md5.update(value.encode("utf-8"))
    md5value=md5.hexdigest()
    if(md5value[:HEX_LEN]!=submd5):
        print ("[-]Access Failed")
        return;
    print ("[*]I am the world smartest CV system!")
    print ("[+]Give me a wing to fly?")
    sys.stdout.flush()
    global input_image_name
    image=sys.stdin.readline()[:-1]
    if(len(image)>200000):
        print("[-]input too long!")
        return;
    input_image_name=tofile(image)
    if input_image_name=="":
        print ("[-]base64 please!")
        sys.stdout.flush()
        return
    if not check():
        print ("[-]You cannot fool me!")
        sys.stdout.flush()
        return
    (input_image_class,input_image_classname)=classify(input_image_name)
    (std_image_class,std_image_classname)=classify(std_image_name)
    if input_image_class!=std_image_class and input_image_class==503:
        print("[*]Wow I get the wing")
        print("[*]Give you the flag")

        print(getflag())
        sys.stdout.flush()
        return
    else:
        print("[*]Give me the wing!")
        sys.stdout.flush()
        return

main()
```

题目原图

简单来说，就是生成对抗样本，扰动率低于0.8且神经网络识别为wing，属于白盒有目标攻击。
本地环境是Anaconda+python3+keras-gpu
在Anaconda里直接装tensorflow-gpu装不上，装keras-gpu就装上了，还带了tensorflow-gpu。
题目提示使用的神经网络是inception-2015-12-05.tgz。
然而题目只用了一句import inception就导入了，找了很久教程，最后找到一个TensorFlow-Tutorials，只要把压缩包里的解压放在一个目录下，就可以一句话导入，第一次使用会自动下载inception-2015-12-05.tgz，下载比较慢，可以自己下好解压放在对应目录下。

一开始是参考这篇文章的TensorFlow 教程 #11 - 对抗样本 - 知乎
使用的应该是FGSM方法，一步步下来确实生成了对抗样本。



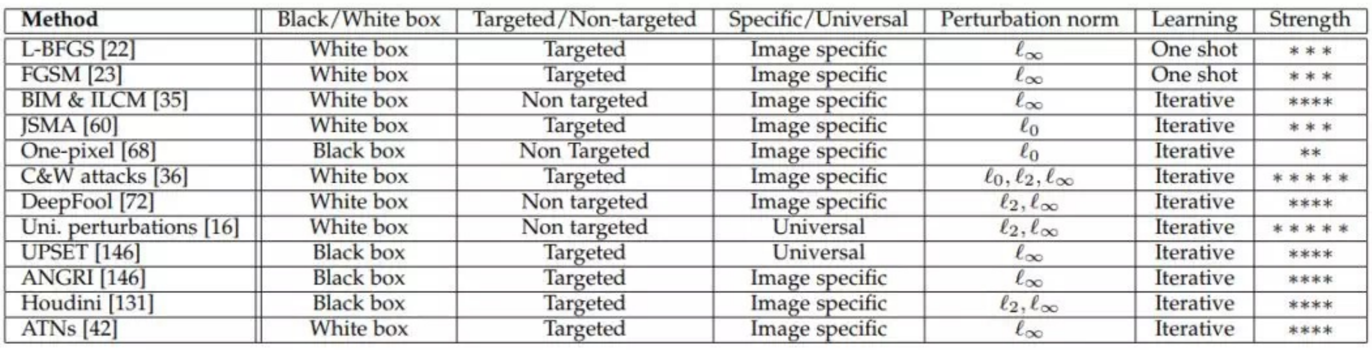
对抗样本1

修改一下这个plt输出的图片，去掉白边黑框。上传上去没拿到flag。

本地查了一下扰动率，得到diff=1.349157168264337，尝试修改各种生成参数，都无法降低扰动率，直到比赛结束。

后续看到有foolbox advbox cleverhans，三个对抗样本库，以后再试一下。

方法综述



Method	Black/White box	Targeted/Non-targeted	Specific/Universal	Perturbation norm	Learning	Strength
L-BFGS [22]	White box	Targeted	Image specific	ℓ_∞	One shot	* * *
FGSM [23]	White box	Targeted	Image specific	ℓ_∞	One shot	* * *
BIM & ILCM [35]	White box	Non targeted	Image specific	ℓ_∞	Iterative	****
JSMA [60]	White box	Targeted	Image specific	ℓ_0	Iterative	* * *
One-pixel [68]	Black box	Non Targeted	Image specific	ℓ_0	Iterative	**
C&W attacks [36]	White box	Targeted	Image specific	$\ell_0, \ell_2, \ell_\infty$	Iterative	* * * * *
DeepFool [72]	White box	Non targeted	Image specific	ℓ_2, ℓ_∞	Iterative	****
Uni. perturbations [16]	White box	Non targeted	Universal	ℓ_2, ℓ_∞	Iterative	* * * * *
UPSET [146]	Black box	Targeted	Universal	ℓ_∞	Iterative	****
ANGRI [146]	Black box	Targeted	Image specific	ℓ_∞	Iterative	****
Houdini [131]	Black box	Targeted	Image specific	ℓ_2, ℓ_∞	Iterative	****
ATNs [42]	White box	Targeted	Image specific	ℓ_∞	Iterative	****

攻击方法对比

对抗攻击概念

目前来说，比较主流的工具有cleverhans，foolbox，另外笔者还发现了一个advertorch，专门针对pytorch模型。

	cleverhans	**foolbox**	**advertorch**
针对模型框架	tensorflow/keras/pytorch	pytorch/tensorflow	pytorch
产生速度	可以批量	无法批量	可以批量
使用复杂度	♥♥♥♥♥	♥♥	♥♥
包含的攻击类别数	♥♥♥♥♥	♥♥♥♥♥	♥♥

（当然这个难易程度是笔者自己分的，见仁见智了~~~）

工具