2021DASCTF八月挑战赛Writeup

原创

citytwilight 于 2021-09-04 14:59:42 发布 357 收藏

分类专栏： CTF 文章标签： python

本文链接：https://blog.csdn.net/weixin_46198176/article/details/120099605

2021DASCTF八月挑战赛Writeup

MISC

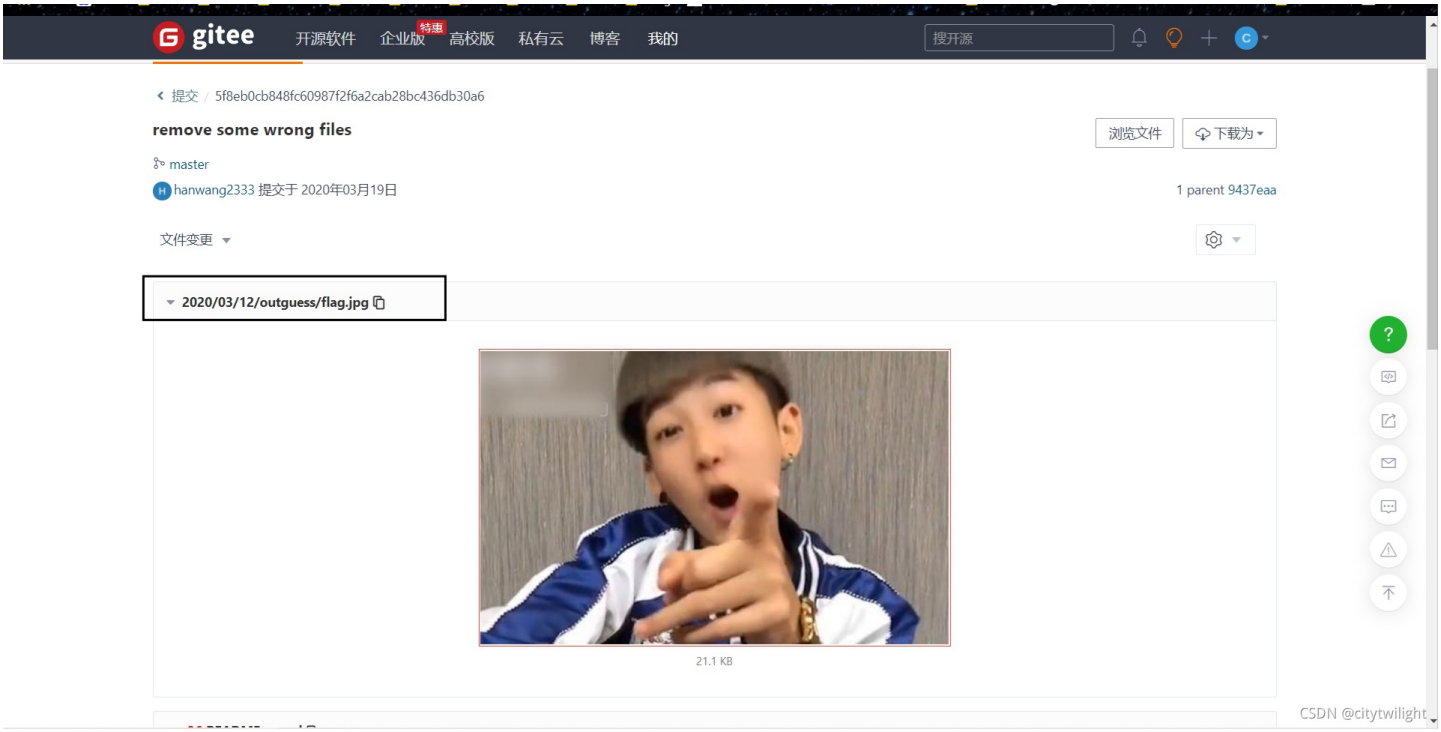
签到

看看公告就有啦(嘿嘿嘿，真-签到题)

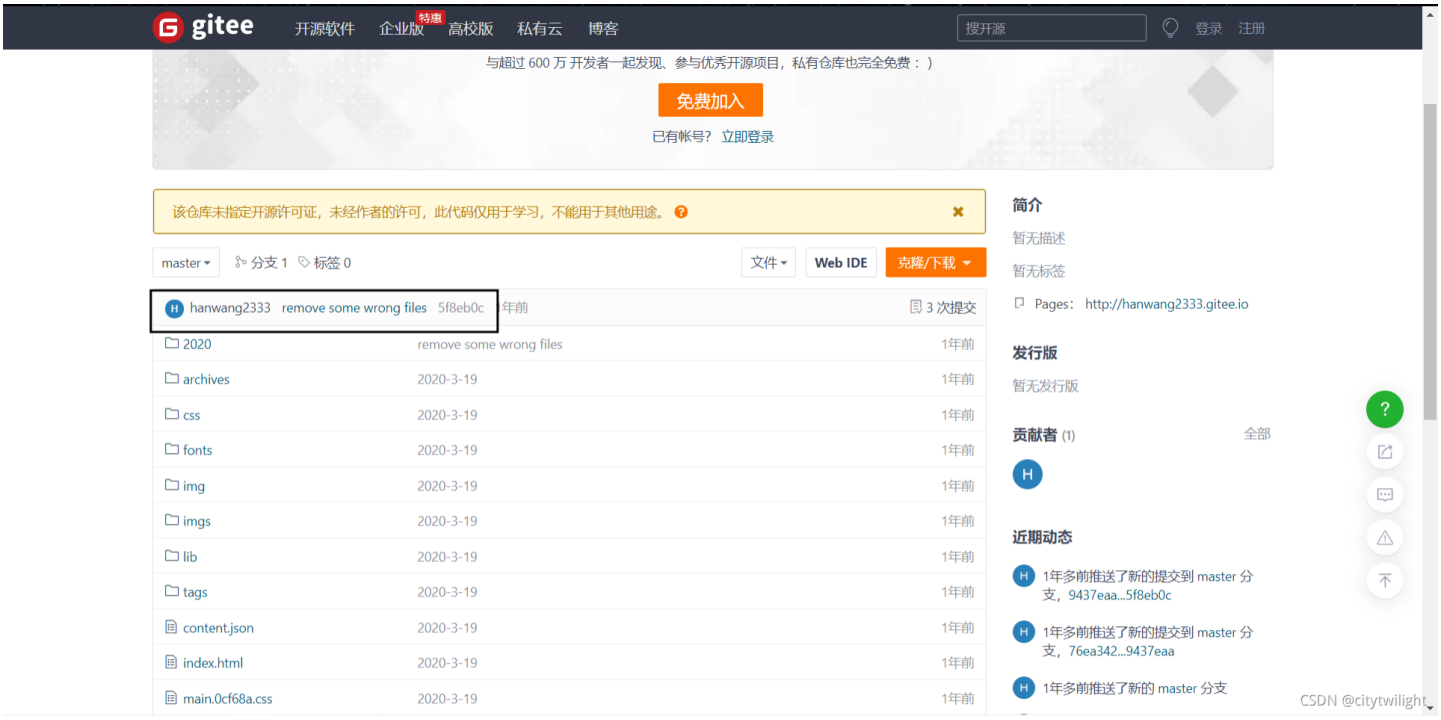
```
flag{welcome_to_dasctf_aug}
```

寒王’sblog

到给的博客里面看，可以看到写了关于outguess隐写的内容，整个博客就博客头像一个图片，猜测是图片，拿博客内容里面的密码试，不对。仔细观察博客，要找到flag.jpg就行了，因为之前有过搭建博客的经验，就决定到这个博客的搭建的平台gitee去寻找，找到gitee上的仓库



找到最新上传的文件，找到flag.jpg



在装有outguess的lunix系统下使用outguess的命令

```
outguess -k "hahahahahahaha" -r flag.jpg out.txt
```

得到flag

```
flag{50aa7fe02602264e7d8102746416cd74}
```

stealer

方法一

打开流量包，过滤DNS，发现有很多重复的数据，过滤ip

```
dns and ip.src==172.27.221.13
```

将info取出，观察发现是图片的base64编码，将字符串进行编辑方便转码。

```
字符串的变化如下：
原字符串：
Standard query 0x6a7a A iVBORw0KGgoAAAANSUhEUgAABMoAAAMxCAIAAACVY8g6AAAAAXNSR0IAr-.s4c6QAAAARnQU1BAACxjwv8YQUAAAAJcEhZcwAADsMAAA7DAcdvqGQAAP-.ISURBVHhe7P1HIBzZmd6NUxtqjrShdtzoHB6ttMROswINp/U9OiGKwd-.f8KbRDtXw3hdQMGWBgjcF1432aDZN04MUySGHMxqvgUajGWg*6fsoffpL-.ctf.com.cn OPT

操作：
1、去除多余字符串”Standard query 0x6a7a A”、”ctf.com.cn OPT”、”-.”
2、将“*”替换为“+”

转化后字符串：iVBORw0KGgoAAAANSUhEUgAABMoAAAMxCAIAAACVY8g6AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAJcEhZcwAADsMAAA7DAcdvqGQAAPISURBVHhe7P1HIBzZmd6NUxtqjrShdtzoHB6ttMROswINp/U9OiGKwdf8KbRDtXw3hdQMGWBgjcF1432aDZN04MUySGHMxqvgUajGWg+6fsoffpL
```

然后拿到图片

方法二(别人的做法，没弄懂)

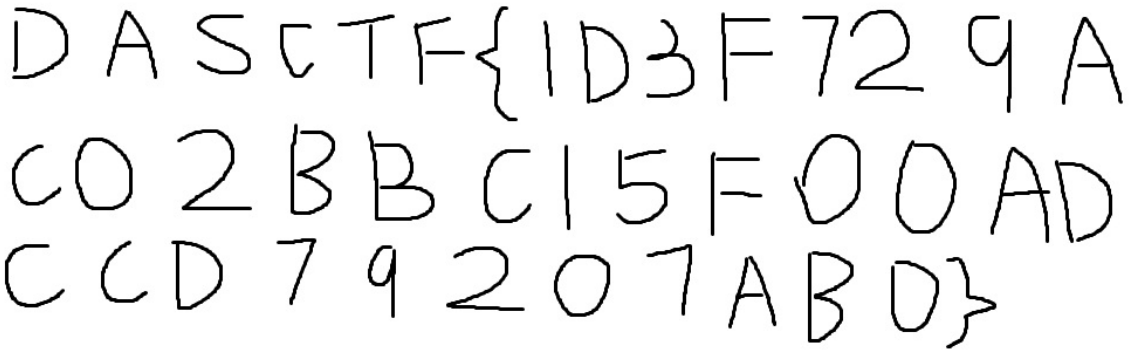
使用tshark命令(linux安装)

```
tshark -r  dump.pcapng -T fields -e dns.qry.name  -Y "udp and dns.flags.response==0"  > res.txt
```

```
from Crypto.Util.number import *
import base64

with open('D:\\桌面\\res.txt','r')as f:
 lines=f.readlines()
tmp=[]
# tmp1=[]
for i,line in enumerate(lines):
 if '-.ctf.com.cn' in line:
  line1=line.strip().replace("-.ctf.com.cn","").replace('-.',"").replace("*","+")
  tmp.append(line1)
  # tmp1.append(len(line1))
b=''.join(tmp)
# print(b)
b1=base64.b64decode(b.encode())
open('res.png','wb').write(b1)
# print(tmp1)
```

拿到图片



flag(有点坑，只要输{}里面的md5就行)（md5里面没有o、l，所以图片里面的就是0和1）

```
1d3f729ac02bbc15f00adccd79207ab0
```

CRYPTO

easymath

题目：

```
assert(len(open('flag.txt', 'rb').read()) < 50)
assert(str(int.from_bytes(open('flag.txt', 'rb').read(), byteorder='big') << 10000).endswith(
    '18627908845631605823658885308696903976675466287107950315443043781547695594104732764822654487543886559810913134195496891693811155735394225459330449025270202092599380954662830008'))
```

解题脚本：

```
import gmpy2
r = 18627908845631605823658885308696903976675466287107950315443043781547695594104732764822654487543886559810913134195496891693811155735394225459330449025270202092599380954662830008
y = r //(2 ** 175)
flag=(y*gmpy2.invert((2 ** 9825),(5 ** 175)))%(5 ** 175)
t=int(flag).to_bytes(50,byteorder='big')
print(t)
```

推理过程看大佬详细的WP

let’s play with rsa~

题目：

```
from sympy import isprime,nextprime
from Crypto.Util.number import getPrime as getprime ,long_to_bytes,bytes_to_long,inverse
flag='flag{***************}'

def play():
    p=getprime(1024)
    q=getprime(1024)

    n=p*q
    e=65537

    print "Hello,let's play rsa~\n"
    print 'Now,I make some numbers,wait a second\n'
    n1=getprime(200)
    n2=getprime(200)
    number=n1*n2
    print "Ok,i will send two numbers to you,one of them was encoded.\n"
    print "Encode n1:%d,\n"%(pow(n1,e,n))
    print "And n2:%d.\n"%n2

    print "Information that can now be made public:the public key (n,e):(%d,%d)\n"%(n,e)
    while True:
        try:
            c=int(raw_input("ok,now,tell me the value of the number (encode it for safe):"))
        except:
            print "Sorry,the input is illeagal, and the integer is accept~"
        else:
            break
    d=inverse(e,(p-1)*(q-1))
    m=pow(c,d,n)
    if m==number:
        print "It's easy and interesting,didn't it?\n"
        print "This is the gift for you :"+flag
    else:
        print "Emmmmm,there is something wrong, bye~\n"

if __name__ == '__main__':
    play()
```

题目给出n、e、pow(n1,e,n)、n2，求c

$c = number \bmod n = (n1*\ldots$

```
a = 12961409879680531683887750211274207457513312084569696075205671038512045983337591135848947097185699604453906986636309421174482607699320759296109315315716510712493223473758909665964696575970829701893860144925111613206750829054110563245759651636500800410800611799341091945238575572204744180794490600917575137957033704047292694479180193784052662788188456092611151703198998693787287001909323881137356776611226888433795185583110831437797736663143463934550216639249647171621868213923613785097805958460050341741859668930711065658867650077543948915545547210086665007538174470035036590197762464220663638774581879789345436101376
n2 = 97565539474126579180941830376626496435261896212953264704896 3
n=17984939304898650803158254840695698991976109768815074321859078838424106854968841553367705813676890566467526226777393337854287622085939699972674300671017434385268427211627149875201145197397614084981293689781191198014025392826496825995772507143839566355038037041002531894155725475951426118850290776380334951977592054772067742756171095427402698085736400771503751576111160577981596580675760373492947131376839865309631659266553436954094643785398989702090587019292927772255810184788705872695851653803767757272932373185881635106333541090761339331432007675440553128966093172385018789133381793617410838502288664808530049842759
e = 65537
c = (a * pow(n2, e, n)) % n
print(c)
```

提交c返回得到flag

ezRSA

题目：

```
from secret import flag
from Crypto.Util.number import *
from random import getrandbits
from hashlib import sha256


class EzRsa:
    def __init__(self):
        self.E = 0x10001
        self.P = getPrime(1024)
        self.Q = getPrime(1024)
        while GCD((self.P-1)*(self.Q-1), self.E) != 1:
            self.Q = getPrime(1024)
        self.N = self.P*self.Q

    def encrypt(self):
        f = getrandbits(32)
        c = pow(f, self.E, self.N)
        return (f, c)

    def encrypt_flag(self, flag):
        f = bytes_to_long(flag)
        c = pow(f, self.E, self.N)
        return c


def proof():
    seed = getrandbits(32)
    print(seed)
    sha = sha256(str(seed).encode()).hexdigest()
    print(f"sha256({seed>>18}...).hexdigest() = {sha}")
    sha_i = input("plz enter seed: ")
    if sha256(sha_i.encode()).hexdigest() != sha:
        exit(0)


if __name__ == "__main__":
```

```
if __name__ == "__main__":
    proof()
    print("welcome to EzRsa")
    print("""
    1. Get flag
    2. Encrypt
    3. Insert
    4. Exit
    """)
    A = EzRsa()
    coin = 5
    while coin > 0:
        choose = input("> ")
        if choose == "1":
            print(
                f"pow(flag,e,n) = {A.encrypt_flag(flag)}\ne = 0x10001")
            exit(0)
        elif choose == "2":
            f, c = A.encrypt()
            print(f"plain = {f}\ncipher = {c}")
            coin -= 1
        elif choose == "3":
            q = getrandbits(1024)
            n = A.P*q
            f = getrandbits(32)
            c = pow(f, 0x10001, n)
            print(f"plain = {f}\ncipher = {c}")
            coin -= 1
        elif choose == "4":
            print("bye~")
        else:
            print("wrong input")
    print("Now you get the flag right?")
```

给你5个coin，1次选择选项1，剩下四次选择选项2、3，当然是平均分配啦。得到四组f、c，两组同q解n，两组不同q解p

2选项
k1n=m21^e-c21
k2n=m22^e-c22
如果gcd(k1,k2)!=1:
那么gcd(k1n,k2n)=k5n

3选项
k3p*q=m31^e-c31
k4p*q=m32^e-c32
如果gcd(k3,k4)!=1:
那么gcd(k3p*q,k4p*q)=k6p

注意！！ 要考虑不互素的情况

解题脚本：

```
import gmpy2
import libnum

m21 = 432540985
c21 = 71661053291461260003468467990916508989669991555598720713341653730858246396433573403877368676739489494372956
45705541881374251463627954867215153754248479607455314501625059368006554456541594829525164594977575051605373580652
73494877195513273612414493148397625403869803790632069153610259514281725463512831654864052674225839603773662497 4
20145277556649874968652112568911707782561467672873788975805113603300486811050686588578099450392236399510368583502
14305862126959421485260880988697376791033172416989217783691336645680449701320745602504926888012562036421449108 9
```

639877735872634910024522894511013457996471379556170467562583 6

m22 = 832433238
c22 = 706807277444487458084341768435286216750957579944704479092509827302280548433963662872605128230836158034580360747064283693824771417467018541884444811039662041564099033678872832676447716106920736794244941700395582831622812060238492098516105772085671372631910226414212482385219125303561477689952817016108152781898007956707194233695330750562742519026826989623292766314220733580329477066817407589664876313254786598679446649096371159253882152724156513243431530920304385088433088047138480457785868712100836625539204096413761561262271824922145137305055003598876243097186690904309350614351483025920567612491441164045494976951 4

m31 = 803092604
c31 = 985806131209365449676896859609750122467861452062588794573317001223602065088460666256752594523072049690839804138470899961821712042445683583752025447279429878280178532242668254003783569090220951850652935517249369301275378313344714196400180190810824672586920342200278299143570574305276424328947494479410564796481057520151632900663162343147694260522316915725241790370389430663603558774549069912535007870520647085600572486757621980620158612484641759989771260769893122797379407707819920885104856938678138376893017463678262352785743429893373805773546861918344562000822385013270863489190749985437499118701152919704223064695 9

m32 = 407827977
c32 = 986351182834512265531841778975832636297215524938052104609436310633091466771954136985316135483263822930050413157997381837398756538123705351433575614886534753494632861246800834691230746662187434863428957960225560954538709897307906349276391290522550572094825092534545815175793259624669275557005137900745262116093615187022630190545445567097562289158957799528802875598239823917863490585837005929600206794328344469846265738101616906825482024886487031535831300572137017763277508712426883716529348857347258550556311417852292662939413184318376148037918725121303495808939734281518735211182075220906408463998849044246608539236 3

c = 738112577675195944638334801658051025770479811984566594520194403305255593565671369846724795923379012732721435542964825563162898530045860874007855595888348991794461743910582781522463083694773376568474775663669832101097713660973867390594562458589365905627117166206901803451070848058560743157324658621114559175932943796792983768354213567435350892005564575713319428839674811270421329448381614780620604319447613128486909873964017154451837328898071048545298982244843199865124852572004590309684318971499258040645344552446634657349692433067812316314549757204318695179504353366344472458895968028467945621729308423400061875902 2
e = 0x10001

# 3选项，求p
def p_def(p):
    k = 1
    while 1:
        if gmpy2.is_prime(p):
            return p
        elif p % (k) == 0 and k != 1:
            p = p // k
        else:
            k += 1

# 2选项求q
def q_def(n, p):
    k = 1
    q = n // p
    while 1:
        if gmpy2.is_prime(q):
            return q
        elif p % (k) == 0 and k != 1:
            q = q // k
        else:
            k += 1

kp = gmpy2.gcd(m31 ** e - c31, m32 ** e - c32)
p=p_def(kp)

```
kn = gmpy2.gcd(m21 ** e - c21, m22 ** e - c22)
#print(kn)
q=q_def(kn,p)
n=p*q

print(n==p*q)
phi=(p-1)*(q-1)
d=gmpy2.invert(e,phi)
flag=pow(c,d,n)
print(flag)
print(libnum.n2s(int(flag)))
```