ctf刷题日记

原创

AndrewMe8211 于 2021-10-02 21:35:58 发布 90 收藏

分类专栏： ctf 文章标签： python

密码学

BUUCTF MD5

给一段MD5加密后的32位码,使用https://www.cmd5.com/工具对这段码进行破解得到flag

> md5加密后是16位或者32位的字符，由字母和数字组成，字母大小写统一；
> 理论上无法解密，除非暴力破解

这里的这个网站就是一个使用暴力破解来解码的网站

BUUCTF 丢失的MD5 1

给一段python源码如下:

```
import hashlib
for i in range(32,127):
    for j in range(32,127):
        for k in range(32,127):
            m=hashlib.md5()
            m.update('TASC'+chr(i)+'O3RJMV'+chr(j)+'WDJKX'+chr(k)+'ZM')
            des=m.hexdigest()
            if 'e9032' in des and 'da' in des and '911513' in des:
                print des
```

我这里显示update函数里面的参数需要加密,修改后是这样的:

```
# Andrew82106
# time: 2021/10/2 21:02
import hashlib
for i in range(32,127):
    for j in range(32,127):
        for k in range(32,127):
            m=hashlib.md5()
            m.update('TASC'.encode('UTF-8')+chr(i).encode('UTF-8')+'O3RJMV'.encode('UTF-8')+chr(j).encode('UTF-8')+'WDJKX'.encode('UTF-8')+chr(k).encode('UTF-8')+'ZM'.encode('UTF-8'))
            des=m.hexdigest()
            if 'e9032' in des and 'da' in des and '911513' in des:
                print(des)
```

运行后得到flag

> hashlib补充

```
>>> import hashlib
>>> m = hashlib.sha256()              # 通过构造函数获得一个hash对象
>>> m.update(b'Nobody inspects')      # 使用hash对象的update方法添加消息
>>> m.update(b' the spammish repetition')  # 同上
>>> m.digest()                # 获得bytes类型的消息摘要
b'\x03\x1e\xdd}Ae\x15\x93\xc5\xfe\\\x00o\xa5u+7\xfd\xdf\xf7\xbcN\x84:\xa6\xaf\x0c\x95\x0fK\x94\x06‘
>>> m.hexdigest()                 # 获得16进制str类型的消息摘要
'031edd7d41651593c5fe5c006fa5752b37fddff7bc4e843aa6af0c950f4b9406'
>>> m.digest_size                 # 查看消息摘要的位长
32
>>> m.block_size                  # 查看消息摘要的内部块大小
64
更简洁的用法：
>>> hashlib.sha224(b"Nobody inspects the spammish repetition").hexdigest()
'a4337bc45a8fc544c03f52dc550cd6e1e87021bc896588bd79e901e2'

hash.update(arg)
更新hash对象。连续的调用该方法相当于连续的追加更新。例如m.update(a); m.update(b)相当于m.update(a+b)。注意，当数据规模较大的时候，Python的GIL在此时会解锁，用于提高计算速度。
一定要理解update()的作用，由于消息摘要是只针对当前状态产生的，所以每一次update后，再次计算hexdigest()的值都会不一样。

hash.digest()
返回bytes格式的消息摘要

hash.hexdigest()
与digest方法类似，不过返回的是两倍长度的字符串对象，所有的字符都是十六进制的数字。通常用于邮件传输或非二进制环境中。通常我们比较摘要时，比较的就是这个值！

hash.copy()
返回一个hash对象的拷贝
```

BUUCTF 一眼就解密

出来的密码看到后面有个等号，想到是不是base64编码。一搞，果然是，就出来了。

base64原理：
https://www.cnblogs.com/luguo3000/p/3940197.html

BUUCTF Url编码 1

使用Url编码解码器就可以了

url是什么：https://blog.csdn.net/houqicun/article/details/78296886

BUUCTF 看我回旋踢

一看就很像凯撒密码，试了一下果然是

BUUCTF 摩丝 1

文件下载下来看发现是摩尔斯电码，解密后套上flag{}得到答案

BUUCTF password

离谱，居然还有这种题。。。姓名首字母+生日=flag

BUUCTF 变异凯撒

这题正确的思路应该是先去找标识符，也就是先假设密码里面的前四个字母对应的是flag，这样的话就可以去找规律。从ascii的角度出发，那么不难知道，这个密码每一位的位移都不一样，第一位在ascii里面移动了5，第二位是6，第三位是7，以此类推，就得到flag

BUUCTF Quoted-printable

使用Quoted-printable揭秘工具就可以得到flag

quoted-printable是什么:http://blog.chacuo.net/494.html

BUUCTF rabbit

rabbit 解密工具即可：http://www.jsons.cn/rabbitencrypt/

至于rabbit是个啥。。。不是很懂

BUUCTF篱笆墙的影子

题名就提示了是栅栏密码，那就对应的解密工具解密就ok了

BUUCTF RSA

就是RSA加密的过程，写个脚本就ok（我一开始居然还把欧拉函数写错了。。。）

```
def gcdInverse(a, n):
    x, y, gcd = Math.myExtGCD(a, n)
    t = 0
    while x<0:
        t = t + 1
        x = x + (int(n/gcd))*t
        y = y - (int(a/gcd))*t
    return x

p = 473398607161
q = 4511491
phi = (p-1)*(q-1)
e = 17
d = gcdInverse(e, phi)
print(d)
```

BUUCTF Alice与Bob

大数分解，自己写了个脚本，然后发现

$O(n),n=98554799767$ 数太大了，就尝试用n=800000来碰运气，估计它的质因子中小的那个是不会超过8000

```
def shai(n):
    for i in range(1, n + 10, 1):
        isprime.append(True)
        if i % 10000 == 0:
            print("cleaning{0}/{1}".format(i,n))
    for i in range(2, n, 1):
        if i % 10000:
            print("shaiing{0}/{1}".format(i,n))
        if isprime[i]:
            prime.append(i)
        for j in prime:
            if j * i > n or j > i:
                break
            isprime[i * j] = False
x = 98554799767
Math.shai(800000)
print("finish shai")
for i in prime:
    print(">>>>>>prime:{0}".format(i))
    if x % i == 0:
        print(i)
        print(int(x/i))
        break
```

BUUCTF rsarsa

很基本的rsa题，对rsa的几个参数(p,q,n,phi(n),e,d)熟悉就可以做出来。根据p,q解出密钥d就可以加密了

ppsucctf Ez_RSA

```
from secret import flag
from Crypto.Util.number import *

N = 256
e = 0x10001
m = bytes_to_long(flag)
p = getPrime(N)
q = getPrime(N)
n = p*q
c=pow(m,e,n)
print("c =",c)
print("n =",n)
# c = 36180247547878328795424029549353365449656959634109576335676307507307869394467692644965994271552510466006031854181508339962773370990693415336694695655716 11
# n = 93832765376080783495356215229917381281873834665584451967474840467210074343109006662850731678529086433418346582208199349494826237477781171197501346289043 01
```

需要大质数分解：https://www.alpertron.com.ar/ECM.HTM

分解后得到p和q，然后就是最基本的模版题了

ppsucctf baby_RSA

```
e=0x10001
m=bytes_to_long(flag)
c=pow(m,e,n)

print("c =",c)
print("n =",n)

#c = 350355553388168485657003148476952468066551745290489044038056534153781757882445957706825007220418294545890702
74312745524192872471746366014877556063351723981035438744039192735063939724595911513278244963033709385633872940 59
#n = 47119307999061849531624878347602604220205747734096755201886348396164153358450342212052892567055446819724391 0
40977771579918043802842183150387194449439904925790307206359905384523125283398643529993103984817917300172010310 90
```

分解n，得到n是由小于等于541的质数相乘得到的，因此可以有计算欧拉函数的式子：

$\phi(n)=$

然后就可以根据题目条件算出d来解密了

BUUCTF RSA1

出现了dp，dq这两个东西，不知道什么意思。先跑个脚本：

```
def dpdq(p, q, dq, dp, c, n):
    i_ = invert(p, q)  # i为p(mod q)的逆元，即p*i = 1(mod q)
    mp = pow(c, dp, p)  # 计算mp = c^dp % p
    mq = pow(c, dq, q)  # 计算mq = c^dq % q

    m = (mp + (i_ * (mq - mp)) * p) % n  # 明文求解公式
    m = hex(m)[2:]  # 转十六进制数据

    flags = ''
    for i in range(len(m) // 2):
        flags += chr(int(m[i * 2:(i + 1) * 2], 16))

    return flags
flag = dpdq(p,q,dq,dp,c,p*q)
```

然后出flag。
原理：

https://blog.csdn.net/xiao_han_a/article/details/118516038?utm_medium=distribute.pc_relevant.none-task-blog-2$_{default}$baidujs_title~default-0.no_search_link&spm=1001.2101.3001.4242.1

Web

- BUUCTF Linux Labs

`ssh -p 25815 root@node3.buuoj.cn` 即可

Misc

- buuctf JING_SAN_PANG

使用stegsolve解析gif得到flag

用之前要配置一手java环境就ok