

XCTF新手区Crypto writeup

原创

KogRow 于 2020-06-30 20:09:53 发布 474 收藏 2

分类专栏: [CTF](#) 文章标签: [CTF crypto](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/shuaicenglou3032/article/details/107044993>

版权



[CTF 专栏收录该内容](#)

59 篇文章 4 订阅

订阅专栏

1.base64

直接拿去base64 decode就行: `cyberpeace{Welcome_to_new_World!}`

2.Caesar

这题凯撒密码。

分别设置位移量为1-25, 输出25条结果后看到位移量为12时有意义, 故位移量为12:

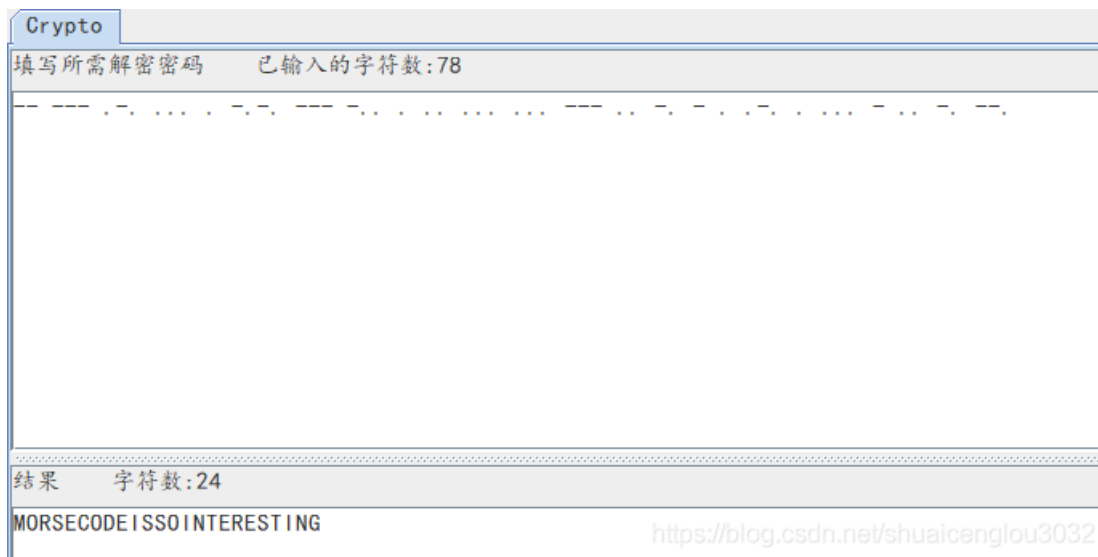
`cyberpeace{you_have_learned_caesar_encryption}`

3.Morse

摩斯密码, 我们在记事本里将1全部替换成“-”, 将0全部替换成“.”, 得到密文:

--- - - - - - - -

摩斯密码解码:

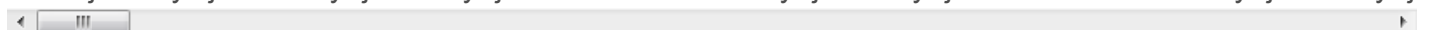


将其转成小写就拿到了flag。

4.混合编码

原文:

`JiM3NjsmlzEyMjsmlzY5OyYjMTlwOyYjNzk7JiM4MzsmIzU2OyYjMTlwOyYjNzc7JiM2ODsmlzY5OyYjMTE4OyYj`



一看就是base64,先解码,解码出来是这样:

```
&#76;&#122;&#69;&#120;&#79;&#83;&#56;&#120;&#77;&#68;&#69;&#118;&#77;&#84;&#65;&#52;&#76;&#122;&#107;&#53;&
```

猜想是unicode的HEX编码,拿去解码:

```
LzEx0S8xMDEvMTA4Lzk5LzExMS8xMDkvMTAxLzExNi8xMTEvOTcvMTE2LzExNi85Ny850S8xMDcvOTcvMTEwLzEwMC8xMDAvMTAxLzEwMi8
```

再次猜想是baae64:

```
/119/101/108/99/111/109/101/116/111/97/116/116/97/99/107/97/110/100/100/101/102/101/110/99/101/119/111
```

这里大佬说是ASCII,把它放进eclipse,将所有“/”替换为“,”,构建数组,转为字符串:

```
public class Main {
    public static void main(String[] args) {
        int[] a = {119,101,108,99,111,109,101,116,111,97,116,116,97,99,107,97,110,100,100,101,102,101,110,99,101,
        String result = "";
        for(int i:a){
            char c = (char)i;
            result+=c;
        }
        System.out.println(result);
    }
}
```

```
<terminated> Main [Java Application] C:\Program Files\Java\jre
welcometoattackanddefenceworld
```

得到结果:

5.幂数加密

密文: 8842101220480224404014224202480122

提示: flag为八位大写字母

这里学习一波幂数加密的知识:

二进制幂数加密法:

二进制幂数加密法,由于英文字母只有26个字母。只要2的0、1、2、3、4、5次幂就可以表示31个单元。通过用二进制幂数表示字母序号数

二进制数除了0和1的表示方法外,在由二进制转换成十进制的时候,还可以表示成2的N次方的形式。例如:

$$15=2^0+2^1+2^2+2^3$$

并且我们发现,任意的十进制数都可以用 2^n 或 $2^n+2^m+\dots$ 的形式表示出来,可以表示的单元数由使用的max n来决定。

$$\text{可表示的单元数}=2^{(n+1)}-1$$

也就是说,根据提示,在0之前分,将密文分为8段: 88421 0122 048 02244 04 0142242 0248 0122

但是如果是2的幂的话只需要0-5就行，密文中不会出现8，因此这和上面说的还有点不太一样。

百度了一下，说是另一种01248密码：

01248密码，又称云影密码...与二进制幂加密不同，这个加密法采用的是0作间隔，其他非0数隔开组合起来相加表示序号1-26之一的字母

上python3代码：

```
#01248密码
def zeronetfe(m):
    alpha = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S',
    m = m.split("0")
    c = ""
    for i in range(0,len(m)):
        str = m[i]
        num = 0;
        for j in range(0,len(str)):
            num += int(str[j])
        c += alpha[num-1]
    return c
def main():
    print(zeronetfe("8842101220480224404014224202480122"))
if __name__ == '__main__':
    main()
```

运行之后就得到结果。

6.Railfence

密文：ccehgyaefnpeoobe{lcirg}epriec_ora_g

这题是栅栏密码，但是和常见的栅栏密码不一样，拿去在工具里解密失败：



根据提示：

到了一看，这个谜面看起来就已经有点像答案了样子了，旁边还画着一张画，是一副农家小院的 图画，上面画着一个农妇在栅栏里面喂5只

暗示栅栏是5。

百度了一下发现是一种叫WWW的变种：

此段介绍来自https://blog.csdn.net/qq_43504939/article/details/98473847

1 2 3 4 5 6 # key=3 Rail-fence Cipher

```
1 . . . 5 . ↘           ↗ ↘
. 2 . 4 . 6           ↘ ↗
. . 3 . . .           ↘ ↗
```

结果为 1 5 2 4 6 3

此处贴一波其他大佬的python加解密代码：

```
def encode(string, key):#需要加密的字符串以及加密栏数
    i = 0
    enlist = []
    for j in range(0, key):
        enlist.append('')#添加分组，列表的一个元素相当于一个分组

    while i < len(string):#分组重排进行加密
        for k in range(0, key):
            if i >= len(string):
                break
            enlist[k] += string[i]
            i += 1
        for k in range(1, key-1):
            if i >= len(string):
                break
            enlist[key-1-k] += string[i]
            i += 1
    enstr = ''
    for i in range(key):
        enstr += enlist[i]
    return enstr
```

代码来自<https://blog.csdn.net/qinying001/article/details/96134356>

```
def decode(string, key):#解密字符串以及解密栏数
    de_key = 2*key - 2#一个部分的长度
    length = len(string)//de_key#确定有多少个完整部分
    r = len(string)%de_key#最后不完整部分的长度
    delist = []
    for i in range(key):
        delist.append('')#重新排布分组
    #确定第一个分组
    if r == 0:
        delist[0] += string[0:length]
        s = length
    else:
        delist[0] += string[0:length+1]
        s = length+1
    #确定第二个到第key-1个分组
    for i in range(1, key-1):
        l = length*2#这几个分组长度至少是完整部分数量的两倍
        #最后一个不完整部分对应当前分组有几个元素
        if r > i:
            l += 1
        if r > de_key-i:
            l += 1
```

```
    delist[i] += string[s:s+1]
    s = s+1
#确定最后一个分组
delist[key-1] += string[s:]
#排布分组确定原字符串
destr = ''
j = 0
for i in range(0, len(string)):
    destr += delist[j][0]
    delist[j] = delist[j][1:]
    if j == key-1:
        flag = 0
    if j == 0:
        flag = 1
    if flag:
        j += 1
    else:
        j -= 1
return destr
```

运行之，拿到明文：

```
cyberpeace{railfence_cipher_gogogo}
Process finished with exit code 0
```

7.easy_RSA

在一次RSA密钥对生成中，假设 $p=473398607161$ ， $q=4511491$ ， $e=17$
求解出 d

这道题简单粗暴，直接上python3代码：

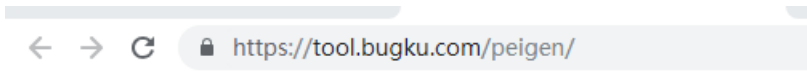
解码:

MAYnullBEnullHAvEnullIAnOTHERnullIDECODEHHHHAAAAABAABBBAABBAAAAAABAABABAAAAAABBBAI

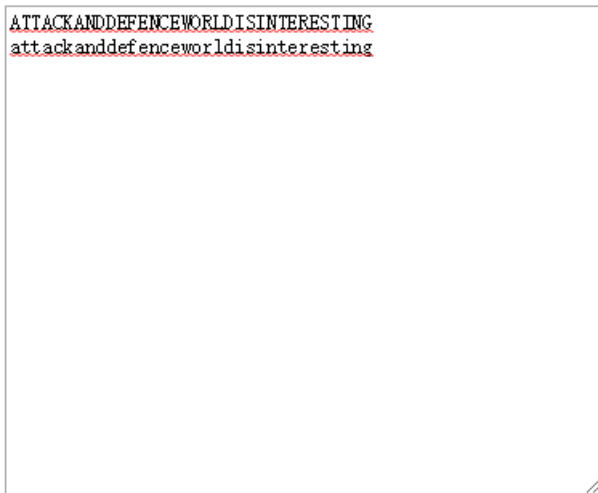
may be have another decode

hhhh AAAAABAABBBAABBAAAAAABAABABAAAAAABBABAAABBAAABBAAABAABAABAABAAABBABAAAE

根据提示，后面这截由A和B组成的估计是培根密码，直接拿来解密，得到:



Bugku|培根密码加解密



<https://blog.csdn.net/shuaicenglou3032>

这里也可以自己写python代码:

https://blog.csdn.net/weixin_42109012/article/details/97644262

9.easychallenge

下载下来是一个pyc文件，使用反编译工具反编译:

```
root@ubuntu /h/pwn# uncompyl6 -o b.py a.pyc
```

得到源代码:

```
# uncompile6 version 3.7.2
# Python bytecode 2.7 (62211)
# Decompiled from: Python 3.5.2 (default, Apr 16 2020, 17:47:17)
# [GCC 5.4.0 20160609]
# Embedded file name: ans.py
# Compiled at: 2018-08-08 20:29:44
import base64

def encode1(ans):
    s = ''
    for i in ans:
        x = ord(i) ^ 36
        x = x + 25
        s += chr(x)

    return s

def encode2(ans):
    s = ''
    for i in ans:
        x = ord(i) + 36
        x = x ^ 36
        s += chr(x)

    return s

def encode3(ans):
    return base64.b32encode(ans)

flag = ''
print 'Please Input your flag:'
flag = raw_input()
final = 'UC7K0WVXWVNKNIC2XCXKHKK2W5NLBKN0USK3LNNVWW3E=== '
if encode3(encode2(encode1(flag))) == final:
    print 'correct'
else:
    print 'wrong'
```

根据源代码编写一个decode方法:


```

import base64
def decode(c):
    mid = base64.b32decode(c)
    mid2 = ''
    for i in mid:
        i = ord(i)^36
        x = chr(i-36)
        mid2 += x
    mid3 = ''
    for i in mid2:
        i = ord(i)-25
        x = chr(i^36)
        mid3 += x
    print mid3
decode('UC7K0WVXWVNKNIC2XCXKHKK2W5NLBKNOUSK3LNNVWW3E===')

```

执行上述代码得到flag:

The screenshot shows an online Python IDE interface. At the top, there is a green button labeled '点击运行' (Click to Run), a dropdown menu set to 'Python 在线工具' (Python Online Tools), and a '清空' (Clear) button. The main editor area contains the following Python code:

```

1 # -*- coding: UTF-8 -*-
2 import base64
3 def decode(c):
4     mid = base64.b32decode(c)
5     mid2 = ''
6     for i in mid:
7         i = ord(i)^36
8         x = chr(i-36)
9         mid2 += x
10    mid3 = ''
11    for i in mid2:
12        i = ord(i)-25
13        x = chr(i^36)
14        mid3 += x
15    print mid3
16 decode('UC7K0WVXWVNKNIC2XCXKHKK2W5NLBKNOUSK3LNNVWW3E===')

```

The output area on the right displays the result: `cyberpeace(interestinghhhhh)`. At the bottom right, there is a URL: <https://blog.csdn.net/shuaicenglou3032>.

10.

11.Normal_RSA

下载下来是一个enc和一个pem文件。

在linux里使用命令：`openssl rsa -pubin -text modulus in warmup -in pubkey.pem`

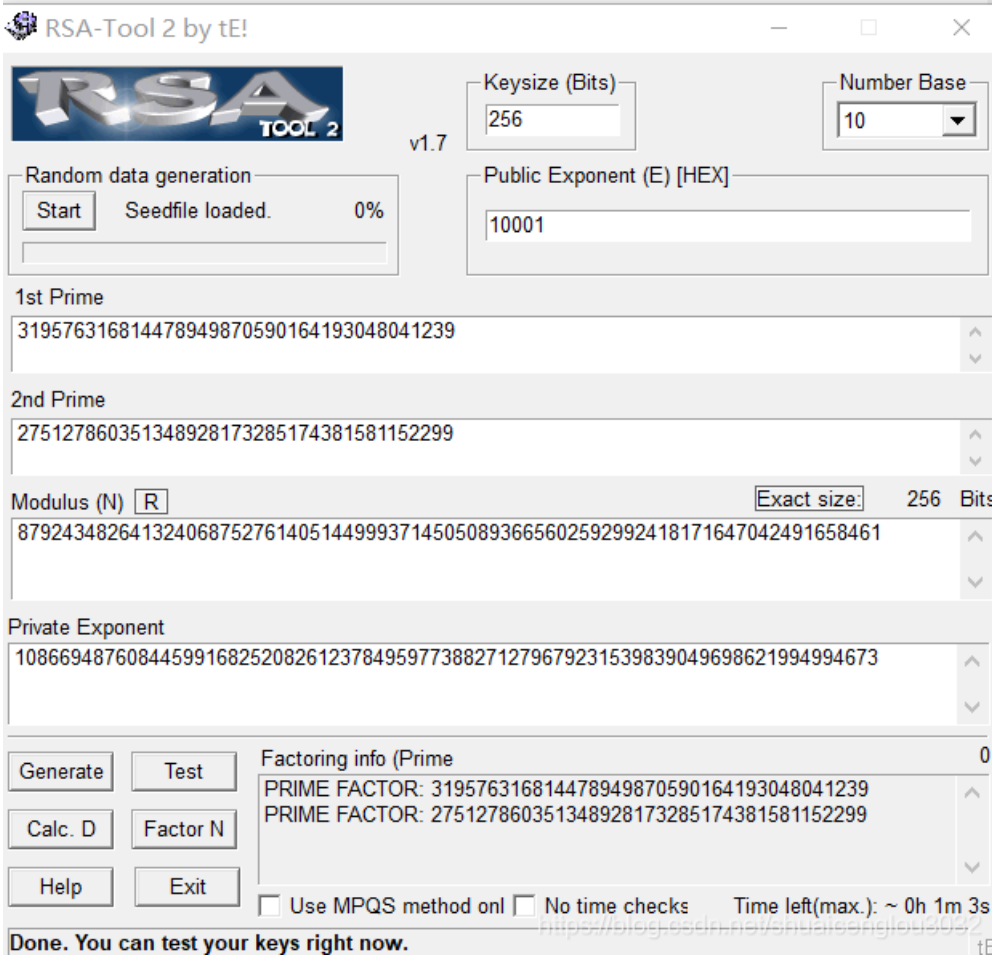
提取公钥：

```

pwn@ubuntu:~$ openssl rsa -pubin -text -modulus -in warmup -in pubkey.pem
Public-Key: (256 bit)
Modulus:
 00:c2:63:6a:e5:c3:d8:e4:3f:fb:97:ab:09:02:8f:
 1a:ac:6c:0b:f6:cd:3d:70:eb:ca:28:1b:ff:e9:7f:
 be:30:dd
Exponent: 65537 (0x10001)
Modulus=C2636AE5C3D8E43FFB97AB09028F1AAC6C0BF6CD3D70EBCA281BFFE97FBE30DD
writing RSA key
-----BEGIN PUBLIC KEY-----
MDwwDQYJKoZIhvcNAQEBBQADKwAwKAIhAMJjauXD20Q/+5erCQKPGqxsC/bNPXDr
yigb/+l/vjDdAgMBAAE=
-----END PUBLIC KEY-----
https://blog.csdn.net/shuaicenglou3032

```

将十六进制的公钥放进RSATool，暴力分解，20分钟之后得到结果：



p:319576316814478949870590164193048041239

q:275127860351348928173285174381581152299

然后点击Calc D，计算私钥d:

10866948760844599168252082612378495977388271279679231539839049698621994994673

然后根据上述生成私钥文件:

```
python rsatool.py -f PEM -o private.pem -p 275127860351348928173285174381581152299 -q 319576316814478949870
```

再使用private.pem解密enc文件:

```
openssl rsautl -decrypt -in flag.enc -inkey private.pem -out flag.txt
```

就能拿到flag

12.easy_ECC

已知椭圆曲线加密Ep(a,b)参数为

p = 15424654874903

a = 16546484

b = 4548674875

G(6478678675,5636379357093)

私钥为

k = 546768

求公钥K(x,y)

使用大佬写的工具ecctool，参数设置如下：

The screenshot shows the ECCTOOL application interface. The 'General Settings' section includes fields for CurveBits (64), ThreadPriority (Normal/8), ecm_n (50), Cost (0s 15ms), NumberBase (10), Seed Padding (Type any chars here), ecm_k (101), CPU (07c14:b60b9d6a), RNG Salt (Pau), and Type (8CABDC779EFE1300E9B45ED34D8FADEB056F80DC1FFB85DF001182BB8CC574325). The CurveType is set to GF(P). The curve equation is Y^2 = X^3 + A*X + B (mod P). The KeyPairs section shows Q[Order] (0), k[Priv] (546768), Gx[Base] (6478678675), Gy[Base] (5636379357093), Rx[Pub] (13957031351290), and Ry[Pub] (5520194834100). The interface also includes buttons for GENERATE, GET NP, GET ORD, GET AB, FACTOR NP, Kangaroo k*G=R, SAVE CFG, LOAD CFG, CLEAR ECC, CLEAR KEY, CHK ORDER, TEST, RAND G, NEW K, NEW G, CALC R, L*G+H*R, CHK Gy, CHK Ry, PAUSE, STOP, ABOUT, and EXIT. A status bar at the bottom shows 'Done Pub: R(x,y) = k * G(x,y)'. A watermark 'https://blog.csdn.net/qq_3032' is visible in the bottom right corner.

点击calc R就得到结果 R(x,y)