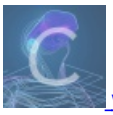
Bugku CTF 加密writeup （未完待续）

原创

KRDecad3 于 2018-05-18 23:40:21 发布 2531 收藏 5

分类专栏： writeup 文章标签： Bugku writeup Crypto

本文链接：https://blog.csdn.net/KRDecad3/article/details/80370443



Bugku CTF 加密writeup

本人CTF小白一枚，此wp是参考网上大佬的wp再加上自己操作写出来的，如有纰漏，还请指正。

0x01滴答~滴

从提供的密码可以猜测是摩尔斯密码，利用bugku自带的解码工具进行解码，
space为空格，short为“.”，long为“-”，得到flag。

0x02聪明的小羊

20

一只小羊翻过了2个栅栏

KYsd3js2E{a2jda}

提示中有“栅栏”，则猜测为栅栏密码，并且栏数为2，利用bugku自带工具解码，得到flag。

0x03ok

ok

30

Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.
Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook! Ook? Ook! Ook! Ook. Ook? Ook. Ook.
Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.
Ook. Ook.
Ook. Ook? Ook. Ook? Ook! Ook. Ook? Ook. Ook. Ook. Ook. Ook! Ook.
Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook. Ook?
Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook! Ook? Ook! Ook! Ook. Ook? Ook! Ook!
Ook! Ook!
Ook! Ook! Ook? Ook. Ook? Ook! Ook. Ook? Ook! Ook! Ook! Ook!
Ook! Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook.
Ook? Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook? Ook! Ook! Ook.
Ook? Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook? Ook. Ook? Ook! Ook. Ook?
Ook. Ook.
Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook. Ook? Ook. Ook. Ook. Ook.
Ook. Ook.
Ook. Ook! Ook? Ook! Ook! Ook. Ook? Ook! Ook! Ook! Ook! Ook!
Ook! Ook? Ook.
Ook? Ook! Ook. Ook? Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook.
Ook! Ook! Ook!
Ook! Ook! Ook! Ook! Ook! Ook! Ook. Ook? Ook. Ook. Ook. Ook. Ook.
Ook. Ook.

Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook? Ook! Ook! Ook

此加密为ook加密，利用ook解码工具，将文本解码，得flag。

0x04这不是摩斯密码

```
+++++ +++++ [->++ +++++ +++<] >++.+ +++++ .<+++ [->-- -<]>- -.+++ +++.<
++++[ ->+++ +<]>+ +++.< +++[- >---< ]>--- .---- .<+++ ++++[ ->--- ----<
]>--- ----- ----- .<+++ ++++[ ->+++ ++++< ]>+++ ++.<+ +++++ +[->- -----
-<]>. <++++ ++++[ ->+++ +++++ <]>++ .<+++ [->-- -<]>- ----. <++++ +++[-
>---- ---<] >---- ----. +++++ +..++ +++.+ .<+++ [->-- -<]>- --.<+ +++++
+[->+ +++++ +<]>+ ++.++ +.+++ +++++ +.--- -.+++ ++.<+ ++[-> +++<] >++++
++.<
```


此加密为brainfuck加密，利用解码工具，得flag。

0x05简单加密

e6Z9i~]8R~U~QHE{RnY{QXg~QnQ{^XVIRXIp^XI5Q6Q6SKY8jUAA

从末尾的AA猜测出可能是凯撒移位和base64的结合，
首先参照ASCII吗，A是65，=是61，偏移4位。用工具中的凯撒移位解码，得到一串base64编码，再进行base64解码，得到flag。（本题的思路需要之前有一定的经验积累）

0x06一段Base64

文件给出了一段base64，先按base64解密，得到如下字符：

```
\134\170\65\143\134\170\67\65\134\170\63\60\134\170\63\60\134\17
\134\170\63\64\134\170\65\143\134\170\67\65\134\170\63\60\134\17
\134\170\63\66\134\170\63\71\134\170\65\143\134\170\67\65\134\17
\134\170\63\60\134\170\63\66\134\170\63\67\134\170\65\143\134\17
\134\170\63\60\134\170\63\60\134\170\63\66\134\170\63\66\134\170
\134\170\67\65\134\170\63\60\134\170\63\60\134\170\63\66\134\170
\134\170\65\143\134\170\67\65\134\170\63\60\134\170\63\60\134\17
\134\170\63\70\134\170\65\143\134\170\67\65\134\170\63\60\134\17
```

通过观察可知是八进制编码，可以将八进制直接转成文本，或者转成“\x”开头的十六进制再转文本，之后得到“\u”开头的Unicode

```
String.fromCharCode
(38,35,120,50,54,59,38,35,120,50,51,59,38,35,120,51,49,59,38,35,120,51
,48,59,38,35,120,51,50,59,38,35,120,51,98,59,38,35,120,50,54,59,38,35,
120,50,51,59,38,35,120,51,49,59,38,35,120,51,48,59,38,35,120,51,56,59,
38,35,120,51,98,59,38,35,120,50,54,59,38,35,120,50,51,59,38,35,120,51,
57,59,38,35,120,51,55,59,38,35,120,51,98,59,38,35,120,50,54,59,38,35,1
20,50,51,59,38,35,120,51,49,59,38,35,120,51,48,59,38,35,120,51,51,59,3
8,35,120,51,98,59,38,35,120,50,54,59,38,35,120,50,51,59,38,35,120,51,5
1,59,38,35,120,51,55,59,38,35,120,51,98,59,38,35,120,50,54,59,38,35,12
0,50,51,59,38,35,120,51,53,59,38,35,120,51,53,59,38,35,120,51,98,59,38
```

编码，在进行Unicode解码，得到如下：
得到的是十进制数，将此十进制数转换成文本，得到HTML实体字符编码，进行解码，出来还是一串HTML实体字符编码，再解一次，就得到flag了（其中%7B和%7D是URL编码，换成大括号）。

0x07.!?

.!?

80

```
..... ..... ..... ..... !?!!. ?..... ..... ..... ..... .??! .?... .!...
..... ..... !.?.. ..... !?!!. ?!!!! !!?.? !.?!! !!!.. ..... ..... .!.?.
..... ...!? !!.?. ..... ..?.? !.?.. ..... .!.?. ..... ..... !?!!. ?!!!!
!!!!! !?.?! .?!.? ..... .....! ?!!.? ..... ...?. ?!.?. ..... !.?.. .....
!?!!. ?!!!! !!?.? !.?!! !!!!! !!!!. ..... ...!. ?..... ...!? !!.?. .....
??!. ?..!. ?..... ..... !?!!. ?!!!! !!!!? .?!.? !!!!! !!!!! !!!.? .....
..!?! !.?.. ....? .?!.? ....! .!!!. !!!!! !!!!! !!!!! !!.?. ..... .!?!!
.?... ...?. ?!.?. ..... ! !!!! !!!!! !.?.. ..... ..!?! !.?.. ..... .??!
.?... ..... !.?.
```


又是ook编码。（ook特点：有问号，点号，感叹号）

0x08+[]-

+[]-

80

```
+++++ +++++ [->++ +++++ +++<] >++.+ +++++ .<+++ [->-- -<]>- -.+++
+++.<
++++[ ->+++ +<]>+ +++.< +++++ [->-- ---<] >.<++ ++[-> ++++< ]>+++
.<+++
[->-- -<]>- ----. ++++. <+++[ ->+++ <]>+. <++++ [->-- --<]> ----- -.<++
+[->+ ++<]> ++.-. ----- ---.< +++[- >+++< ]>+++ .---- .<+++ [->-- -<]>-
.<+++ +++[- >---- --<]> ----- ----. +.<++ +++++ +[->+ +++++ ++<]> +++++
+++++ .<
```


又是brainfuck编码。（brainfuck特点：有加号，减号，方括号，尖括号）

0x09奇怪的密码

奇怪的密码

100

突然天上一道雷电
gndk€rlqhmtkwwp}z

从密码中看到一个欧元符号和一个右花括号，分析前四个字母，其ASCII码为
103，110，100，107；flag四个字母的ASCII码为
102，108，97，103，可以看出，第一个向前移一位，第二个向前移两位，以此类推。
写一个简单解码脚本（PHP写不来，最后用C语言写的）

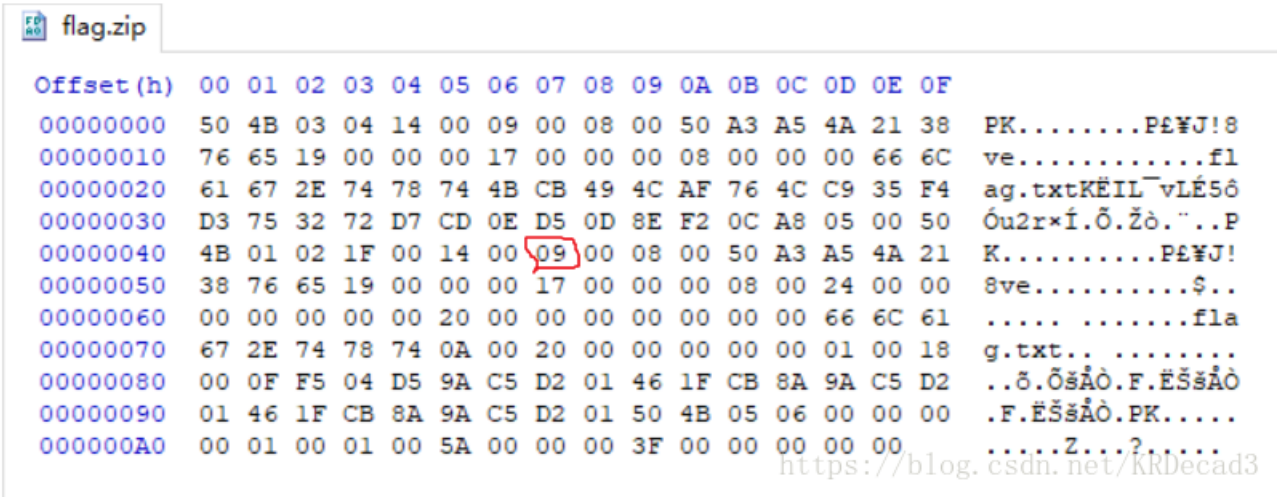
```
#include <stdio.h>
#include <string.h>

int main(void)
{
 int i, tmp = 1, len;
 char str[] = "gndk€rlqhmtkwwp}z";

 len = strlen(str);

 for (i = 0; i < len; i++)
 {
  printf("%c", (char)((int)str[i] - tmp));
  tmp++;
 }

 return 0;
}
```

0x10托马斯.杰斐逊

这题用到的是关于托马斯杰斐逊的转轮密码。按照密钥数字先将密码本的每一行重新排列，比如密码本的第2行为解码本的第一行，密码本的第5行为解码本的第二行。。。
然后根据密文对每一行进行移位，密文第一个字母为H，就将解码本第一行循环左移至以H开头，成为：HGVSFUWIKPBELNACZDTRXMJQOY，以此类推。
观察每一列，发现倒数第六列有BUGKU字样，那一列就是flag。

0x11zip伪加密

先附上一个关于zip伪加密的讲解：https://blog.csdn.net/u011377996/article/details/79286958（侵权立删）

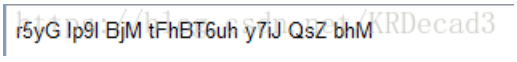


用HxD打开压缩文件，把文件目录区的全局方式位标记中09改成00或其他偶数即可解压，得到flag.txt。

0x12告诉你个秘密(ISCCCTF)

从编码可以猜测出是十六进制，进行ASCII码转换，得到一串可能为base64的编码，

再进行base64解码，得到四个或三个字母为一组的编码，



（此处脑洞有些大）观察键盘发现，四个或三个字母围起来的字符就是解码的字符，例如：r5yG围的字母“T”，等等。
最后的flag格式网上说是flag:xxxxxxx
但我试过很多种，很多遍没成功，等会儿再试试，不过解题思路是这样。

0x13来自宇宙的信号

来自宇宙的信号

110

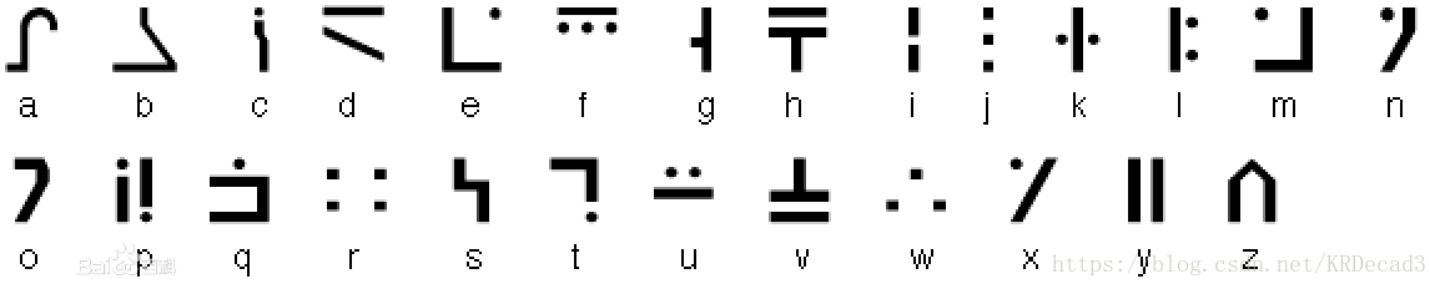
银河战队出击

flag格式 flag{字母小写}

2017102118...

这道题挺有意思的。
打开看到一串奇怪的字符，

到百度上搜搜“银河战队”没搜出来相关的文字，再搜一下“银河文字”，找到相关的文字转换，这种文字叫“标准银河字母”，



然后对照解码，得到flag。

0x14散乱的密文

散乱的密文

60

lf5{ag024c483549d7fd@@1}
一张纸条上凌乱的写着2 1 6 5 3 4

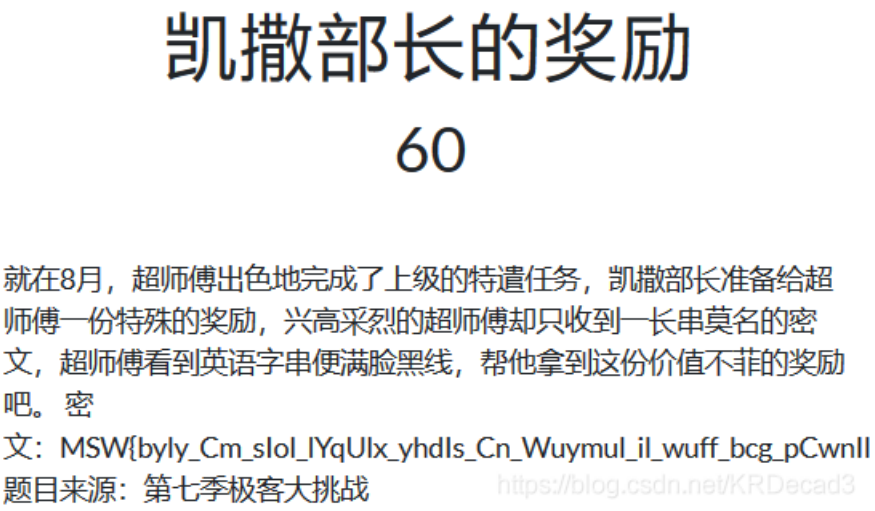
题目给的密文刚开始看以为是维吉尼亚密码，后来仔细想想不是，并且题目说了“散乱”的密码，将密文按顺序排列在数字下方正好排4行，那么就是列移位加密。

2	1	6	5	3	4

2	1	6	5	3	4
l	f	5	{	a	g
0	2	4	c	4	8
3	5	4	9	d	7
f	d	@	@	1	}

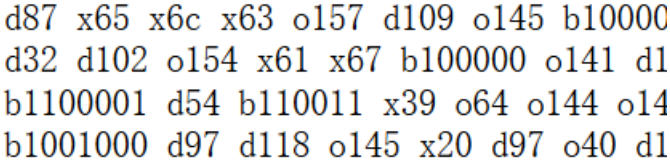
按照数字顺序一列一列写下来得到
f25dl03fa4d1g87}{c9@544@
再进行一次栏数为2的栅栏密码解密，得到flag。
@@是填充字符，不属于flag的内容。

0x15凯撒部长的奖励



这道题就是一个简单的凯撒密码，因为出自极客大挑战，flag头就是SYC，知道这个就好推算移位数了。直接用工具就能跑出来。

0x16 进制转换



这里可以看出就是把这些二进制b八进制o十进制d十六进x制转换成字符，一个一个查不太容易，可以写脚本完成。
附上一个网上看到的Java脚本（自己编程菜）

```
import java.util.Scanner;

public class test1 {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc=new Scanner(System.in) ;
        while(true){
            String str=sc.nextLine();
            String b="",o="",d="",x="";
```

```
            //System.out.println(str.length());
            for(int i=0;i<str.length();i++)
            {
                if(str.charAt(i)=='b')
                {
                    b="";
                    while(str.charAt(++i)!=' ')
                    {
                        b+=str.charAt(i);

                    }
                    int ten = Integer.parseInt(b, 2);
                    //System.out.println(ten);
                    System.out.print((char)ten);
                }
                else if(str.charAt(i)=='o')
                {
                    o="";
                    while(str.charAt(++i)!=' ')
                    {
                        o+=str.charAt(i);

                    }
                    int ten = Integer.parseInt(o, 8);
                    //System.out.println(ten);
                    System.out.print((char)ten);
                }
                else if(str.charAt(i)=='d')
                {
                    d="";
                    while(str.charAt(++i)!=' ')
                    {
                        d+=str.charAt(i);
                    }
                    int ten = Integer.parseInt(d, 10);
                    //System.out.println(ten);
                    System.out.print((char)ten);
                }
                else if(str.charAt(i)=='x')
                {
                    x="";
                    while(str.charAt(++i)!=' ')
                    {
                        x+=str.charAt(i);
                    }
                    //System.out.println(x);
                    int ten = Integer.parseInt(x, 16);
                    //System.out.println(ten);
                    System.out.print((char)ten);
                }
            }
        }
    }
}
```

0x17 affine

affine

100

y = 17x-8 flag{szzyfimhyzd}
答案格式：flag{*}

来源：第七届山东省大学生网络安全技能大赛

由题可知这是仿射密码，加密算法为`y=(17x-8) mod 26`，
那么解密算法就是`x=((17的逆)y+8) mod 26`。
求逆公式就是：`17*(17re)mod26=1`
下面给出解密脚本：

```
# x-求17的逆：
x = 1
while (17*x%26 != 1):
    x += 1

chiper = "szzyfimhyzd"
plain = ""
for i in range(len(chiper)):
    plain += chr(x*(ord(chiper[i])-ord('a')+8)%26+ord('a'))
print(plain)
```

0x18 rsa

```
N :
46065781388428960989637205658554417248531811702624626389974432923749270182062721955600778820059011913617389598900138215153600685382332638289236314360431451868638878600298924880081486124859507532627709964533869497709745916853089877600729369572810197606942397169652423775522718706141820284991147912479399072259782789919834535022875373093108939327546391654445662689424541509610783446577840953237318712531855461472259930179152891621283936812106603554100880826153450058602365276771227162578520428096468800468032830012484968047710530251937737009257810782711682139182621097232037761496754782761989571260259486856902729020438686670354441296247148207862836064657849735343618207098163901787287368569768472521344635567334299356760080507454640207003
```

题目给了N，e，密文，其中n和e相差不大，在RSA中e很大的情况下，可以使用wiener attack进行破解，这里使用一个RSA破解工具 `RsaCtfTool.py` 。

安装方法

```
root@kali:~/RsaCtfTool# python RsaCtfTool.py --createpub -n 46065781388428960989637205658554417248531811702624626389974432923749270182062721955600778820059011913617389598900138215153600685382332638289236314360431451868638878600298924880081486124859507532627709964533869497709745916853089877600729369572810197606942397169652423775522718706141820284991147912479399072259782789919834535022875373093108939327546391654445662689424541509610783446577840953237318712531855461472259930179152891621283936812106603554100880826153450058602365276771227162578520428096468800468032830012484968047710530251937737009257810782711682139182621097232037761496754782761989571260259486856902729020438686670354441296247148207862836064657849735343618207098163901787287368569768472521344635567334299356760080507454640207003827116821391826210972320377614967547827619 > test/test.pem
```

```
root@kali:~/RsaCtfTool# python RsaCtfTool.py  --publickey test/test.pem --private > test/test.key
root@kali:~/RsaCtfTool# python RsaCtfTool.py  --key test/test.key --dumpkey
[*] n: 46065781388428960989637205658554417248531811702624626389974432923749270182062721955600778820059011913617389598900138215153600685382332638289236314360431451868638878600298924880081486124859507532627709964533869497709745916853089877600729369572810197606942397169652423775522718706141820284991147912479399072259782789919834535022875373093108939327546391654445662689424541509610783446577840953237318712531855461472259930179152891621283936812106603554100880826153450058602365276771227162578520428096468800468032830012484968047710530251937737009257810782711682139182621097232037761496754782761989571260259486856902729020438686670354441296247148207862836064657849735343618207098163901787287368569768472521344635567334299356760080507454640207003
[*] e: 35461110244130757205657218182792589919834535022875373093108939327546391654445662689424541509610783446577840953237318712531855461472259930179152891621283936812106603554100880826153450058602365276771227162578520428096468800468032830012484968047710530251937737009257810782711682139182621097232037761496754782761989571260259486856902729020438686670354441296247148207862836064657849735343618207098163901787287368569768472521344635567334299356760080507454640207003
[*] d: 82646679722942750172933397723717833221688221494719768342210823934093636918895
[*] p: 15991846970993213322072626901560749932686325766403404864023341810735319249066370916090640926219079368845510444031400322229147771682961132420481897362843199
[*] q: 28805791771260259486856902729020438686670354441296247148207862836064657849735343618207098163901787287368569768472521344635567334299356760080507454640207003
https://blog.csdn.net/KRDecad3
```

使用工具得到p,q,d，再用脚本解密即可：

```python
# -*-coding:utf-8-*-
from libnum import n2s,s2n

# 求最大公约数
def gcd(a, b):
    if a < b:
        a, b = b, a
    while b != 0:
        temp = a % b
        a = b
        b = temp
    return a

# 扩展欧几里得算法
def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b%a, a)
        return (g, x-(b//a)*y, y)

def modinv(a, m):
    g, x, y = egcd(a,m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x%m

if __name__ == '__main__':
    p = 15991846970993213322072626901560749932686325766403404864023341810735319249066370916090640926219079368845
5104440314003222291477716829611324204818973628431 99
    q = 28805791771260259486856902729020438686670354441296247148207862836064657849735343618207098163901787287368
5697684725213446355673342993567600805074546402070 03
    e = 3546111024413075720565721818279258991983453502287537309310893932754639165444566268942454150961078344657 7
840953237318712531855461472259930179152891621283936812106603554100880826153450058602365276771227162578520428096 4
688004680328300124849680477105302519377370092578107827116821391826210972320377614967547827619
    d = 82646679722942750172933397723717833221688221494719768342210823934093636918 95
    # d = modinv(e, (p-1)*(q-1))
    c = 3823099131622939965182356759069230106004462041219173776463238468054625622845151823884296522139471184833 7
832459443844446889468362154188214840736744657885858943810177675871991111466653158257191139605699916347308294995 6
645302808168504827405306022545591237591211063383592202426377759190269335633260694494243911 92
    n = p * q
    m = pow(c, d, n)
    print n2s(m)
```