bugku ——加密 做题记录

原创

Captain Hammer 于 2019-08-18 19:37:28 发布 2977 收藏 8
分类专栏： CTF题解

本文链接：https://blog.csdn.net/vhkjhwbs/article/details/98762710

目录

23 rsa

24 来自宇宙的信号（标准银河字母）

1，滴答—滴

发现一段摩斯密码：-... -.- -.-. - ..-. -- .. ... -.-.

直接在线解码得到：BKCTFMISC

flag： KEY{BKCTFMISC}

2,聪明的小羊

打开发现一段栅栏密码：KYsd3js2E{a2jda}

在线解码得到： KEY{sad23jjdsa2}

3，ok

打开搜了一下 这是一种名叫OOk 的编程语言

解码地址：http://tool.bugku.com/brainfuck/?wafcloud=1

找了个在线解码得到flag：flag{ok-ctf-1234-admin}

4这不是摩斯密码

下载文件打开是一段字符：查询得知是Brainfuck语言，

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

在线解码：http://tool.bugku.com/brainfuck/?wafcloud=1

flag{ok-c2tf-3389-admin}

5，easy_crypto

打开发现是长度不一样的由0和1 组成的字符串，

```
0010 0100 01 110 1111011 11 11111 010 000 0 001101 1010 111 100 0 001101 01111 000 001101 00 10 1 0 010 0 0
```

果断采用莫斯电码解码：

```
flag{m0rse_code_1s_interest1n9!}
```

6，简单加密

看到密文的最后两个字符为 AA：猜想可能是凯撒密码和base64 的结合

base64 的结尾是 == ，而= 的ASCII为 61　A 的 ASCII 为 65，可知凯撒密码的偏移量为 4

把 密文 的每个字符的 ASCII 减去 4 得到的密文 为 base64 的密文，在进行base64解码：

```
import base64


text = input("请输入题目密文")
text1=''
for i in text:
    t=chr(ord(i)-4)
    text1+=t

print(base64.b64decode(text1))
```

得到flag： key{68743000650173230e4a58ee153c68e8}

7，散乱的密文

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

那就在纸上写呗：

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	}

然后按123456得到一个字符串：f25dl03fa4d1g87}{c9@544@

在进行栅栏密码解码：flag{52048c453d794df1}@@

flag： flag{52048c453d794df1}

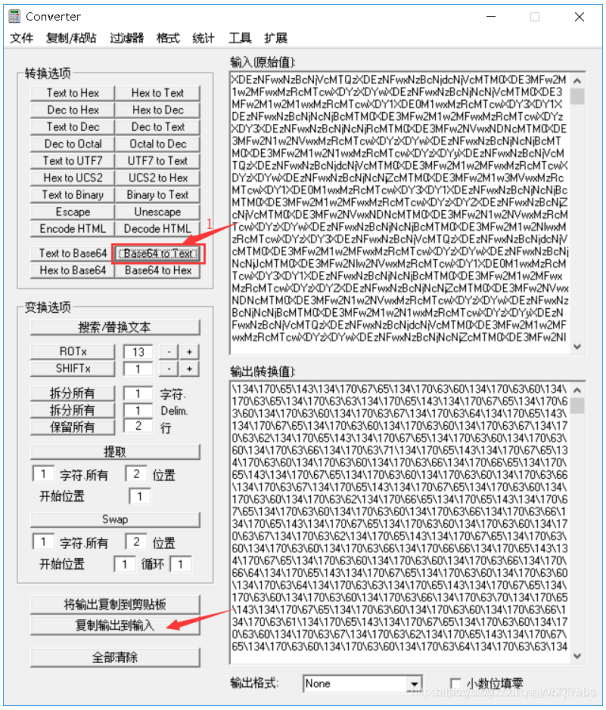
8 凯撒部长的奖励

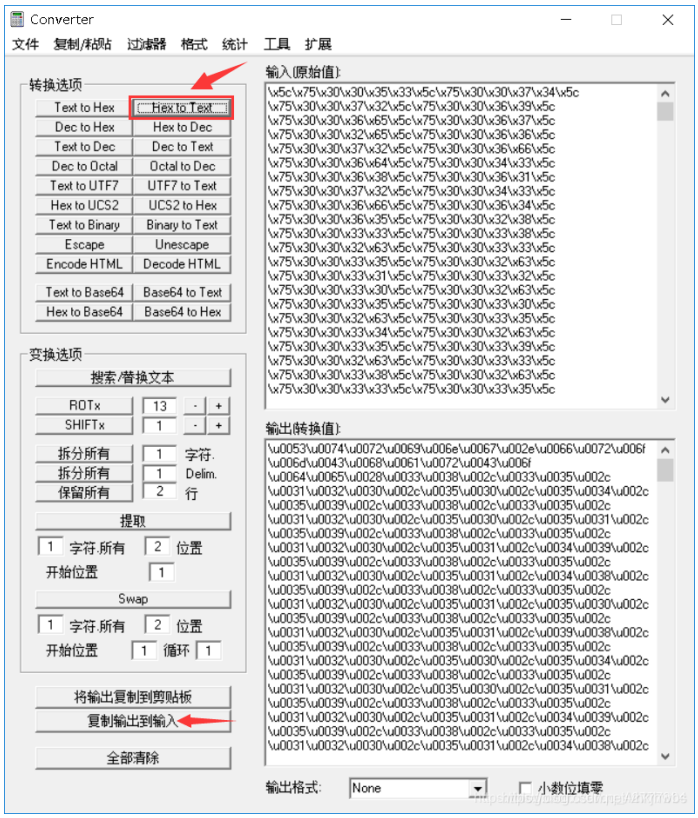
就在8月，超师傅出色地完成了上级的特遣任务，凯撒部长准备给超师傅一份特殊的奖励，兴高采烈的超师傅却只收到一长串莫名的密文，超师傅看到英语字串便满脸黑线，帮他拿到这份价值不菲的奖励吧。 密文：
MSW{byly_Cm_slol_lYqUlx_yhdls_Cn_Wuymul_il_wuff_bcg_pCwnll_cm_u_Yrwyffyhn_guh_cz_sio_quhn_ni_a
题目来源：第七季极客大挑战

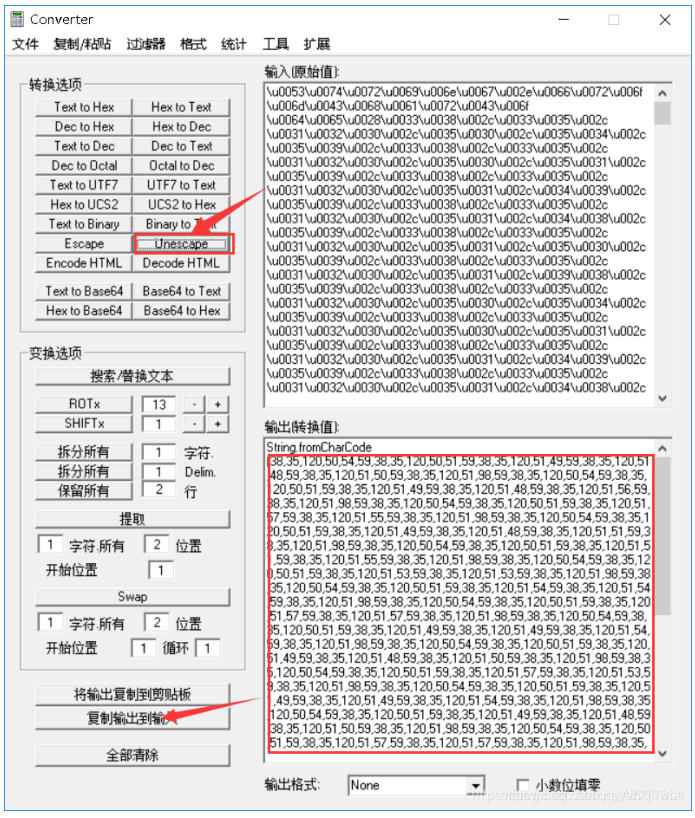
找一个凯撒密码的在线解码，解密后得到：
SYC{here_ls_yOur_rEwArd_enjOy_lt_Caesar_or_call_him_vlctOr_is_a_Excellent_man_if_you_want_to_get_hi

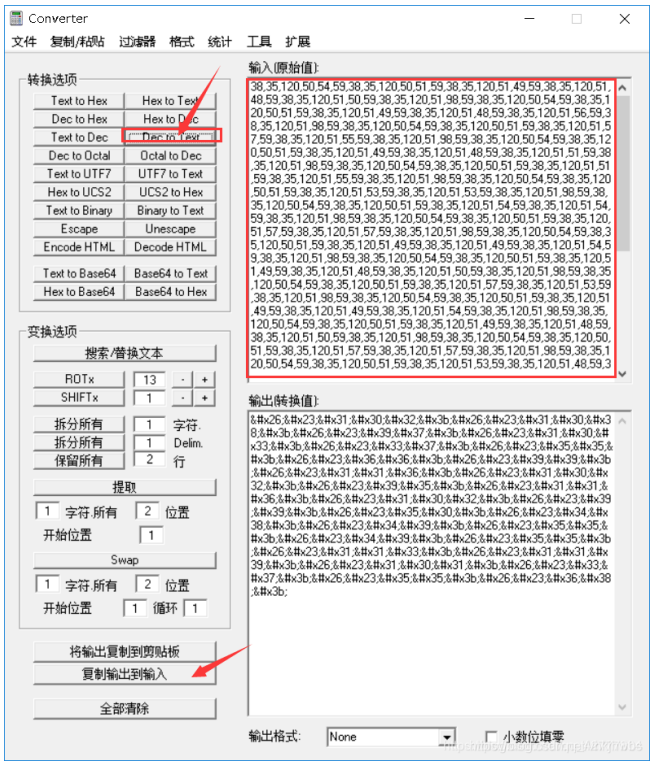
9，一段base64

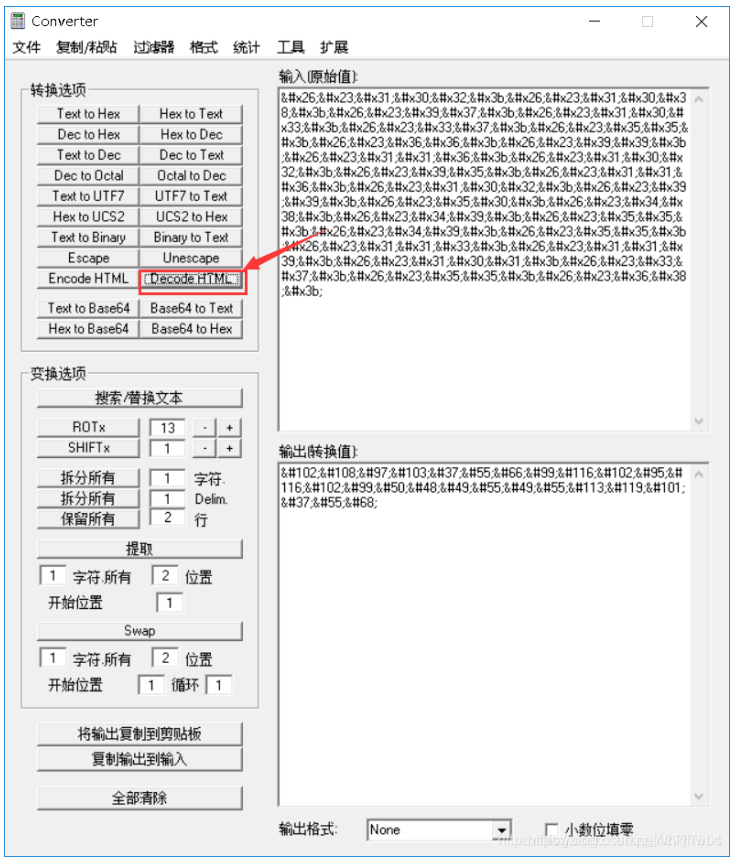
把这一串密文 放入 Converter




Converter
文件 复制/粘贴 过滤器 格式 统计 工具 扩展
转换选项
Text to Hex
Hex to Text
Dec to Hex
Hex to Dec
Text to Dec
Dec to Text
Dec to Octal
Octal to Dec
Text to UTF7
UTF7 to Text
Hex to UCS2
UCS2 to Hex
Text to Binary
Binary to Text
Escape
Unescape
Encode HTML
Decode HTML
Text to Base64
Base64 to Text
Hex to Base64
Base64 to Hex
变换选项
搜索/替换文本
ROTx
SHIFTx
拆分所有
字符.
Delim.
保留所有
行
提取
字符.所有
位置
开始位置
Swap
循环
将输出复制到剪贴板
复制输出到输入
全部清除
输入(原始值):
输出(转换值):
输出格式:
None
小数位填零


Converter
文件 复制/粘贴 过滤器 格式 统计 工具 扩展
转换选项
Text to Hex
Hex to Text
Dec to Hex
Hex to Dec
Text to Dec
Dec to Text
Dec to Octal
Octal to Dec
Text to UTF7
UTF7 to Text
Hex to UCS2
UCS2 to Hex
Text to Binary
Binary to Text
Escape
Unescape
Encode HTML
Decode HTML
Text to Base64
Base64 to Text
Hex to Base64
Base64 to Hex
变换选项
搜索/替换文本
ROTx
SHIFTx
拆分所有
字符.
Delim.
保留所有
行
提取
字符.所有
位置
开始位置
Swap
循环
将输出复制到剪贴板
复制输出到输入
全部清除
输入(原始值):
输出(转换值):
String.fromCharCode
输出格式:
None
小数位填零





Unicode编码 UTF-8编码 URL编码/解码 Unix时间戳 Ascii/Native编码互转

flag%7Bctf_tfc201717qwe%7D

flag%7Bctf_tfc201717qwe%7D

flag{ctf_tfc201717qwe}

10,.!?

这段密文其实是 OOK 语言的变形

在线解码：
flag{bugku_jiami}

11，+[]-（Brainfuck）

一段Brainfuck语言，在线解码：flag{bugku_jiami_23}

12，奇怪的密码

gndk€rlqhmtkwwp}z

对比一下： f l a g 的ascii 为102 108 97 103

g n d k 的ASCII为103 110 100 107

依次 推移 1 2 3 4

根据这个规律 写一个脚本：

```
text = "gndk€rlqhmtkwwp}z"
n =1
t = ''
for i in text:
    t += chr(ord(i)-n)
    n += 1
print(t)
```

得到：flag{lei_ci_jiami}

13托马斯.杰斐逊（杰斐逊转盘加密）

这个转盘加密，比如第一个密钥匙：2、密文匙：H

把转盘第二行单独提出来 2： <KPBELNACZDTRXMJQOYHGVSFUWI <

从H的地方一直剪切，把剪切的内容放在最前面，变成 2： <HGVSFUWIKPBELNACZDTRXMJQOY <

依次类推把14行都按这样的方式整一遍就得到这个：

2： <HGVSFUWIKPBELNACZDTRXMJQOY <
5： <CPMNZQWXYIHFRLABEUOTSGJVDK <
1： <BVIQHKYPNTCRMOSFEZWAXJGDLU <
3： <TEQGYXPLOCKBDMAIZVRNSJUWFH <
6： <SLOQXVETAMKGHIWPNYCJBFZDRU <
4： <XQYIZMJWAORPLNDVHGFCUKTEBS <
9： <WATDSRFHENYVUBMCOIKZGJXPLQ <
7： <CEONJQGWTHSPYBXIZULVKMRAFD <
8： <RJLXKISEFAPMYGHBQNOZUTWDCV <
14：<QWXPHKZGJTDSENYVUBMLAOIRFC <
10：<GOIKFHENYVUWABMCXPLTDSRJQZ <
13：<LTDENQWAOXPYVUIKZGJBMCSRFH <
11：<ENYSRUBMCQWVJXPLTDAOIKFZGH <
12：<SWAYXPLVUBOIKZGJRFHENMCQTD <

我这是是格式问题不太好看出flag，在记事本中容易看出来一点，flag在倒数第六列。

手工太慢 直接上代码：

```
char = "ZWAXJGDLUBVIQHKYPNTCRMOSFE<<KPBELNACZDTRXMJQOYHGVSFUWI<<BDMAIZVRNSJUWFHTEQGYXPLOCK<<RPLNDVHGFCUKTEB

miyao = [2,5,1,3,6,4,9,7,8,14,10,13,11,12]
miwen = "HCBTSXWCRQGLES"

char_array = char.split('<<')
i = 0
result = ""
for n in miyao:
    str = char_array[n-1]
    s = miwen[i]

    postion = str.index(s) # 找到 密文字符在 字符串中的位置
    str1 = str[:postion]   # 截取前半部分字符串
    str2 = str[postion:] # 截取后后半部分字符串
    new_str = str2 + str1  #生成新的字符串
    i +=1
    result += new_str[-6]  #这里是根据输出的字符串 看到倒数第 6 列 是flag 才决定提取每一行的 倒数第6个字符
    print(new_str + "\n")

print("flag{" + result.lower() +  "}") #输出flag
```

得到flag{xsxsbugkuadmin}

14 zip伪加密

首先简单了解一下什么是 zip伪加密

zip 伪加密：

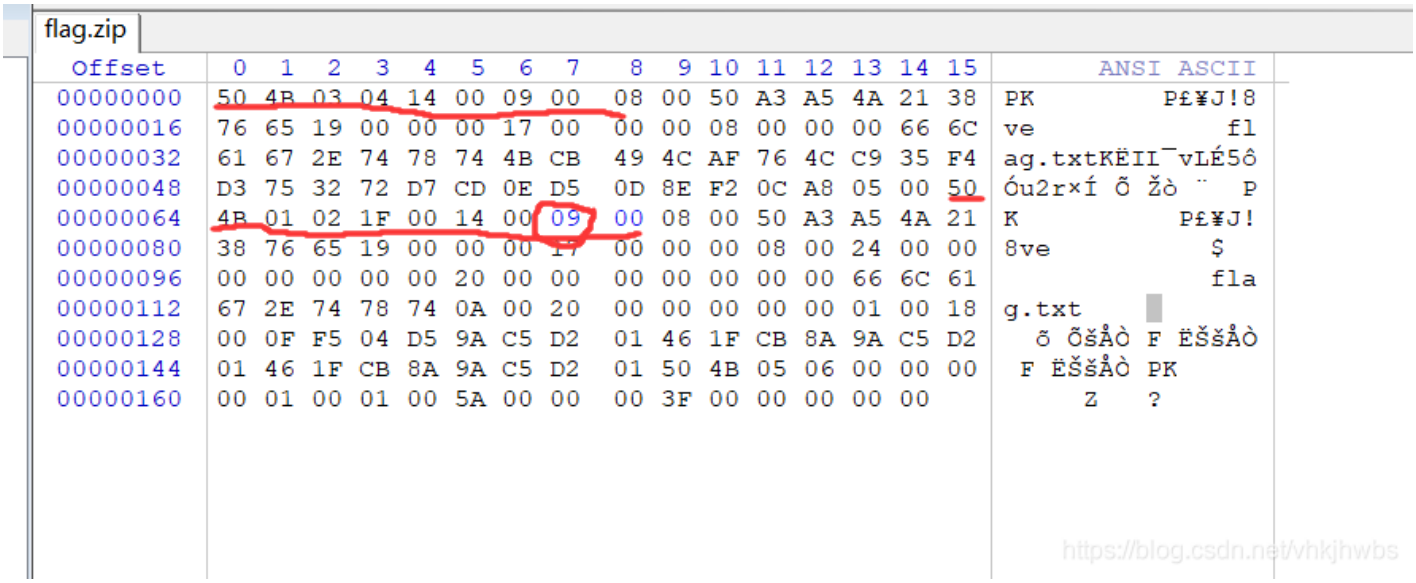
Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	50	4B	03	04	14	00	01	00	08	00	5A	7E	F7	46	16	B5
00000010	80	14	19	00	00	00	17	00	00	00	07	00	00	00	6B	65
00000020	79	2E	74	78	74	0B	CE	CC	75	0E	71	AB	CE	48	CD	C9
00000030	C9	57	28	CE	CC	2D	C8	49	AD	28	4D	AD	05	00	50	4B
00000040	01	02	3F	00	14	00	09	00	08	00	5A	7E	F7	46	16	B5
00000050	80	14	19	00	00	00	17	00	00	00	07	00	24	00	00	00
00000060	00	00	00	00	20	00	00	00	00	00	00	00	6B	65	79	2E
00000070	74	78	74	0A	00	20	00	00	00	00	00	01	00	18	00	65
00000080	58	F0	4A	1C	C5	D0	01	BD	EB	DD	3B	1C	C5	D0	01	BD
00000090	EB	DD	3B	1C	C5	D0	01	50	4B	05	06	00	00	00	00	01
000000A0	00	01	00	59	00	00	00	3E	00	00	00	00	00			

压缩源文件数据区：
50 4B 03 04：这是头文件标记（0x04034b50）
14 00：解压文件所需 pkware 版本
00 00：全局方式位标记（有无加密）
08 00：压缩方式
5A 7E：最后修改文件时间
F7 46：最后修改文件日期
16 B5 80 14：CRC-32校验（1480B516）
19 00 00 00：压缩后尺寸（25）
17 00 00 00：未压缩尺寸（23）
07 00：文件名长度
00 00：扩展记录长度
6B65792E7478740BCECC750E71ABCE48CDC9C95728CECC2DC849AD284DAD0500
压缩源文件目录区：
50 4B 01 02：目录中文件文件头标记(0x02014b50)
3F 00：压缩使用的 pkware 版本
14 00：解压文件所需 pkware 版本
00 00：全局方式位标记（有无加密，这个更改这里进行伪加密，改为09 00打开就会提示有密码了）
08 00：压缩方式
5A 7E：最后修改文件时间
F7 46：最后修改文件日期
16 B5 80 14：CRC-32校验（1480B516）
19 00 00 00：压缩后尺寸（25）
17 00 00 00：未压缩尺寸（23）
07 00：文件名长度
24 00：扩展字段长度
00 00：文件注释长度
00 00：磁盘开始号
00 00：内部文件属性
20 00 00 00：外部文件属性
00 00 00 00：局部头部偏移量
6B65792E7478740A00200000000000010018006558F04A1CC5D001BDEBDD3B1CC5D001BDEBDD3B1CC5
压缩源文件目录结束标志：
50 4B 05 06：目录结束标记
00 00：当前磁盘编号
00 00：目录区开始磁盘编号
01 00：本磁盘上纪录总数
01 00：目录区中纪录总数
59 00 00 00：目录区尺寸大小
3E 00 00 00：目录区对第一张磁盘的偏移量
00 00：ZIP 文件注释长度

如果把一个zip文件的文件头或者加密标志位进行适当修改，那就可能会改变文件的可读性了呗，这样就达到了zip伪加密的目的，可是这里又有一个疑问了，你可能会问，为什么改成09，而不是其他的数字呢，其实改成09只是举的一个例子，只要末位是奇数，就代表加密，反之，末位是偶数代表未加密

本题得到一个 压缩包：flag.zip

用 winhex 打开：



把第压缩源文件目录区的 加密标志位 改为 偶数 就是将 09 改为 00 就可以了

得到flag：
flag{Adm1N-B2G-kU-SZIP}

15 告诉你个秘密(ISCCCTF)

得到一段密文：

636A56355279427363446C4A49454A7154534230526D6843
56445A31614342354E326C4B4946467A5769426961453067

细看可以发现 都是 在 0~F 之内 推测是16 进制 解码后得到：

cjV5RyBscDlJlEJqTSB0RmhCVDZ1aCB5N2lKlFFzWiBiaE0g

猜测可能是 base64 ， 用 base 64 解密后得到：

```
r5yG lp9I BjM tFhBT6uh y7iJ QsZ bhM
```

然后最奇葩的出现了

你绝对想不到 居然是 键盘上 被周围的几个键包围 的 字符：

比如 在 键盘上 r5yg 所包围的 键 是 T

以此类推 得到：

```
TONGYUAN
```

直接提交就可以 **是不是有点变态！！！！！**

16，这不是 md5（16进制）

测一下长度 发现是 44 个字符 肯定不是 md5 啦，既然提到md5 我想可能跟md5 有点 关系

那试一下 sha1（），base64 发现不行

那尝试一下 16 进制转字符 得到：

flag{ae73587ba56baef5} 这是什么啊。。。

17，贝斯家族（base）

谐音吗？？ base 应该是吧

但不知道是base 中的那种编码方式：

64？ 16？ 32？ 58？ 62？85？ 91？ 92？

这题竟然是 base 91 我靠第一次碰到

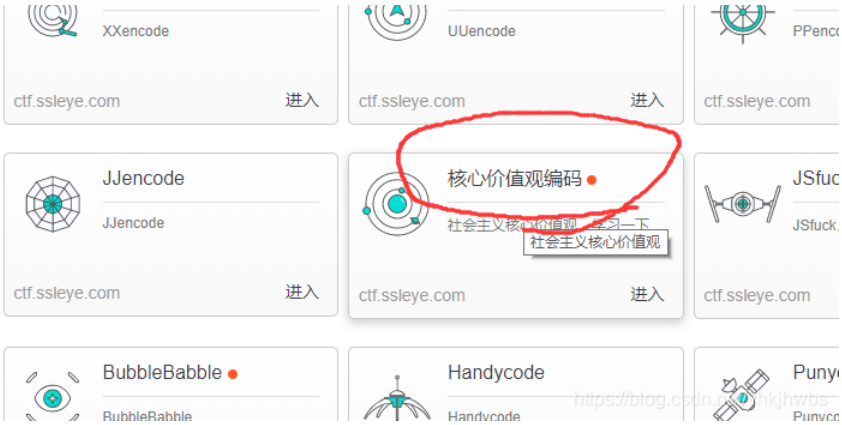
flag{554a5058c9021c76}

附上 工具箱连接：http://ctf.ssleye.com/

18，富强民主（核心价值观密码）

我服了 还有这种 密文

这是一种 叫 核心价值观 的 编码方式



工具箱连接：http://ctf.ssleye.com/

解密得到：
flag{90025f7fb1959936}

19，python(N1CTF)

是两个Python脚本 一个是封装好的 自定义类 N1ES.py 一个是定义类的一个对象

只需要读懂 这 个加密的 类 的加密的方法 相对应写出解密函数

找了一个脚本：（这里只需要修改 N1ES类中 的 解密方法）

```
# -*- coding: utf-8 -*-
import base64
def round add(a,b):
```

```
 f = lambda x,y: x + y - 2 * (x & y)
 res = ''
 for i in range(len(a)):
  res += chr(f(ord(a[i]),ord(b[i])))
 return res

def permutate(table,block):
 return list(map(lambda x: block[x], table))

def string_to_bits(data):
 data = [ord(c) for c in data]
 l = len(data)*8
 result = [0] * l
 pos = 0
 for ch in data:
  for i in range(0,8):
   result[(pos<<3)+i] = (ch>>i) & 1
  pos += 1
 return result

s_box = [54, 132, 138, 83, 16, 73, 187, 84, 146, 30, 95, 21, 148, 63, 65, 189, 188, 151, 72, 161, 116, 63,

def generate(o):
 k = permutate(s_box,o)
 b = []
 for i in range(0,len(k),7):
  b.append(k[i:i+7]+[1])
 c = []
 for i in range(32):
  pos = 0
  x = 0
  for j in b[i]:
   x += (j<<pos)
   pos += 1
  c.append((0x10001**x) % (0x7f))
 return c

class N1ES:
 def __init__(self,key):
  if (len(key) != 24 or isinstance(key,bytes) == False):
   raise Exception("key must be 24 bytes long")
  self.key = key
  self.gen_subkey()

 def gen_subkey(self):
  o = string_to_bits(self.key)
  k = []
  for i in range(8):
   o = generate(o)
   k.extend(o)
   o = string_to_bits([chr(c) for c in o[0:24]])
  self.Kn = []
  for i in range(32):
   self.Kn.append(map(chr,k[i*8: i*8+8]))
  return

 def decrypt(self,plaintext):
  res = ''
  for i in range(len(plaintext)/16):
```

```
    block = plaintext[i*16:(i + 1)*16]
    L = block[:8]
    R = block[8:]
    for round_cnt in range(32):
     L,R = R, (round_add(L, self.Kn[31-round_cnt]))
    L,R = R,L
    res += L + R
   return res


key = "wxy191iss00000000000cute"
nles = N1ES(key)
flag = base64.b64decode("HRlgC2ReHW1/WRk2DikfNBo1dl1XZBJrRR9qECMNOjNHDktBJSxcI1hZIz07YjVx")
flag = nles.decrypt(flag)
print (flag)
```

运行得到 flag ：

```
N1CTF{F3istel_n3tw0rk_c4n_b3_ea5i1y_s0lv3d_/--/}
```

20 进制转换

这里发现是 二进制、八进制、十进制、和十六进制 的混合编码 ， 需要统一化 再转化为 字符

当然是不可能手工转化的

写了一个脚本（Python 中没有 switch/case 的用法 、进制转化时需要将完整形式写出来）

```
start_char = "d87 x65 x6c x63 o157 d109 o145 b100000 d116 b1101111 o40 x6b b1100101 b1101100 o141 d105 x62
char = start_char.split(' ')#将字符串分割为一个数组 方便处理每一个数据
result = []
for c in char:
    if(c[0]=='d'): # 十进制
        result.append(c[1:])
    elif(c[0]=='x'): #16进制
        c = '0' + c
        result.append(int(c,16))
    elif(c[0]=='o'):#8进制
        c = '0' + c
        result.append(int(c,8))
    elif(c[0]=='b'):#2进制
        c = '0' +c
        result.append(int(c,2))

for i in result: #将十进制转化为字符
    print(chr(int(i)),end="")
```

得到：

```
Welcome to kelaibei. Give you a flag as a gift.  flag{1e4bf81a6394de5abc005ac6e39a387b} . Have a good time~
```

flag{1e4bf81a6394de5abc005ac6e39a387b}

21 affine（仿射）

题中 y = 17x-8 是加密算法

flag中的是密文 flag{szzyfimhyzd} 把 szzyfimhyzd 解密

查了一下 仿射密码，七说八说的一大推 懒得看 找了个在线解码：不太好用

看了别人的博客：

写的不错 可以看一看https://blog.csdn.net/weixin_43858318/article/details/89202645

得到

flag{affineshift}

22, Crack it(Linux shadow 文件解密)

没有后缀名的文件 直接放进 winhex看一下：

得到：

```
root:$6$HRMJoyGA$26FIgg6CU0bGUOfqFB0Qo9AE2LRZxG8N3H.3BK8t49wGlYbkFbxVFtGOZqVIq3qQ6k0oetDbn2aVzdhuVQ6US.:177
```

不知道是什么东西

看了一下别人的博客

说是 Linux 中的 shadow文件 shadow文件是被加密过的

kali 中的 John 可以破解：（我没操作，这里粘别人的过程）

```
root@kali:~# john shadow
Warning: detected hash type "sha512crypt", but the string is also recognized as
"crypt"
Use the "--format=crypt" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Press 'q' or Ctrl-C to abort, almost any other key for status
hellokitty       (root)
1g 0:00:00:07 DONE 2/3 (2019-04-05 21:21) 0.1351g/s 678.7p/s 678.7c/s 678.7C/s i
lovegod..celtic
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```



```
root@kali:~# john --show shadow
root:hellokitty:17770:0:99999:7:::
1 password hash cracked, 0 left
```

解密出来居然是 hellokitty

flag{hellokitty}

23 rsa

rsa 加密 我也不太懂 这里找到了 一个 大佬的博客 ：

https://blog.csdn.net/qq_40657585/article/details/84874073

按照大佬的writeup 操作一番

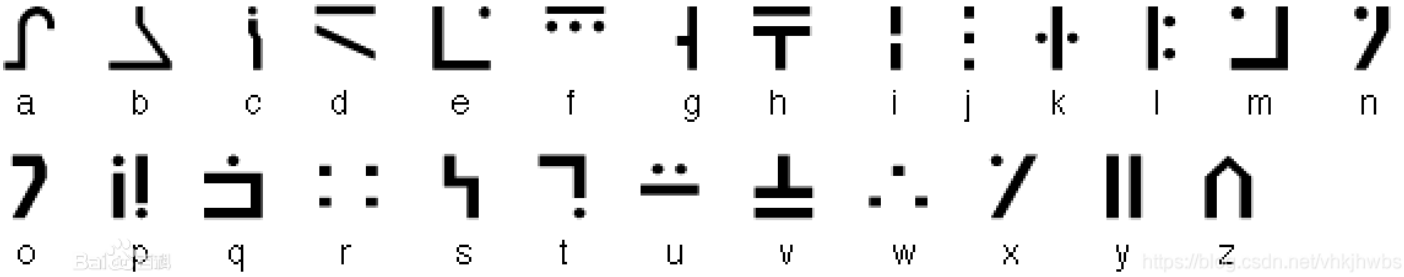
最后得到flag

flag{Wien3r_4tt@ck_1s_3AsY}

24 来自宇宙的信号（标准银河字母）

查了一下是一种叫 标注银河字母 的 语言

对照一下：



得到： nopqrst

得到flag ： flag{nopqrst}