Bugku-加密-writeup

原创

Flemington、 于 2020-04-28 15:23:19 发布 546 收藏

文章标签： CTF 加密

本文链接：https://blog.csdn.net/Everywhere_wwx/article/details/105814662

easy_crypto

```
0010 0100 01 110 1111011 11 11111 010 000 0 001101 1010 111 100 0 001101 01111 000 001101 00 10 1 0 010 0 000 1
01111 10 11110 101011 1111101
```

莫斯电码 0010—>...-.

```
from __future__ import print_function
a = input("input the string:")
s = a.split(" ")
dict = {'01': 'A',
        '1000': 'B',
        '1010': 'C',
        '100':'D',
        '0':'E',
        '0010':'F',
        '110': 'G',
        '0000': 'H',
        '00': 'I',
        '0111':'J',
        '101': 'K',
        '0100': 'L',
        '11': 'M',
        '10': 'N',
        '111': 'O',
        '0110': 'P',
        '1101': 'Q',
        '010': 'R',
        '000': 'S',
        '1': 'T',
        '001': 'U',
        '0001': 'V',
        '011': 'W',
        '1001': 'X',
        '1011': 'Y',
        '1100': 'Z',
        '01111': '1',
        '00111': '2',
        '00011': '3',
        '00001': '4',
        '00000': '5'
```

```
        '10000': '6',
        '11000': '7',
        '11100': '8',
        '11110': '9',
        '11111': '0',
        '001100': '?',
        '10010': '/',
        '101101': '()',
        '100001': '-',
        '010101': '.',
        '110011':',',
        '011010':'@',
        '111000':':',
        '101010':':',
        '10001':'=',
        '011110':"'",
        '101011':'!',
        '001101':'_',
        '010010':'"',
        '10110':'(',
        '1111011':'{',
        '1111101':'}'
        };
for item in s:
    print (dict[item],end='')
#print (s)
```

简单加密

Ascii位移后base64

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

```
import base64

text = input("please input: ")
text1=''
for i in text:
    t=chr(ord(i)-4)
    text1+=t

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

散乱的密文

lf5{ag024c483549d7fd@@1}
一张纸条上凌乱的写着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	}

按照上面的顺序读出来:
f25dl03fa4d1g87}{c9@544@
然后再栅栏解出来下面的密码：
flag{52048c453d794df1}@@
删除@@即为密码。

凯撒部长的奖励

`MSW{byly_Cm_sIol_lYqUlx_yhdIs_Cn_Wuymul_il_wuff_bcg_pCwnIl_cm_u_Yrwyffyhn_guh_cz_sio_quhn_ni_ayn_bcm_chzilgunchm_sio_wuh_dich_om}`

偏移量为+6

`SYC{here_Is_yOur_rEwArd_enjOy_It_Caesar_or_call_him_vIctOr_is_a_Excellent_man_if_you_want_to_get_his_informations_you_can_join_us}`

一段Base64

base64->8进制解密->16进制->unicode->dextoascii->html decode->html decode

flag{ctf_tfc201717qwe}

.!?

brainfuck/Ook

flag{bugku_jiami}

+[]-

brainfuck/Ook

flag{bugku_jiami_23}

奇怪的密码

```
strs= input()
a = 1
for char in strs:
 print(chr(ord(chr)-a))
 a += 1
```

flag{mfjdjkjbnj}

托马斯.杰斐逊

```
# 私钥
key = "2,5,1,3,6,4,9,7,8,14,10,13,11,12"
# 密文
cipher_text = "HCBTSXWCRQGLES"

f = open("zhuanlun.txt")
str_first_encry = []

for line in f:
    line = line.strip()
    str_first_encry.append(line)

key_index = key.split(",")
str_second_encry = []
for k in key_index:
    str_second_encry.append(str_first_encry[int(k) - 1])
    # print (str_first_encry[int(k) - 1])

for i, ch in enumerate(cipher_text):
    line = str_second_encry[i]
    split_index = line.index(ch)
    # print (split_index)
    temp = []
    temp[0:len(line) - split_index + 1] = line[split_index:len(line)]
    temp[len(temp):] = line[0:split_index]
    str_second_encry[i] = "".join(temp)

print ("---------------------------------------------")
for plain in str_second_encry:
    print (plain)
```

flag{xsxsbugkuadmin}

zip伪加密

zip伪加密，50 4B 之后改标志位为 14 00 00 00

flag{Adm1N-B2G-kU-SZIP}

告诉你一个秘密

```
In [1]: import binascii
In [2]: binascii.unhexlify('636A56355279427363446C4A49454A7154534230526D6843
   ...: 56445A31614342354E326C4B4946467A5769426961453067')
Out[2]: b'cjV5RyBscDlJIEJqTSB0RmhCVDZ1aCB5N2lKIFFzWiBiaE0g'
In [3]: import base64
In [4]: base64.b64decode('cjV5RyBscDlJIEJqTSB0RmhCVDZ1aCB5N2lKIFFzWiBiaE0g')
Out[4]: b'r5yG lp9I BjM tFhBT6uh y7iJ QsZ bhM '
```

对应 键盘 = 》 TONGYUAN

这不是md5

```
In [5]: binascii.unhexlify('666c61677b6165373335383762613536626165663 57d')
Out[5]: b'flag{ae73587ba56baef5}'
```

贝斯家族

base91解密

flag{554a5058c9021c76}

富强民主

核心主义价值观编码

flag{90025f7fb1959936}

python(N1CTF)

Feistel加密结构、 DES算法、AES算法

```
import base64, string, N1ES

key = "wxy191iss00000000000cute"
c = base64.b64decode("HRlgC2ReHW1/WRk2DikfNBo1dl1XZBJrRR9qECMNOjNHDktBJSxcI1hZIz07YjVx")
n1es = N1ES.N1ES(key)
f = ""
for i in xrange(3):
    for j in xrange(16):
        for k in string.printable:
            s = "x" * i * 16 + "x" * j + k + "x" * (48 - i * 16 - j - 1)
            e = n1es.encrypt(s)
            check = c[i * 16 + j + 8] == e[i * 16 + j + 8] if j < 8 else c[i * 16 + j - 8] == e[i * 16 + j - 8]
            if check:
                f += k
                break
print f
```

N1CTF{F3istel_n3tw0rk_c4n_b3_ea5i1y_s0lv3d_/–/}

进制转换

```
import binascii

s = ["d87", "x65", "x6c", "x63", "o157", "d109", "o145", "b100000", "d116", "b1101111", "o40", "x6b", "b1100101"
,
     "b1101100", "o141", "d105", "x62", "d101", "b1101001", "d46", "o40", "d71", "x69", "d118", "x65", "x20",
     "b1111001", "o157", "b1110101", "d32", "o141", "d32", "d102", "o154", "x61", "x67", "b100000", "o141", "d11
5",
     "b100000", "b1100001", "d32", "x67", "o151", "x66", "d116", "b101110", "b100000", "d32", "d102", "d108", "d
97",
     "o147", "d123", "x31", "b1100101", "b110100", "d98", "d102", "b111000", "d49", "b1100001", "d54", "b110011"
, "x39",
     "o64", "o144", "o145", "d53", "x61", "b1100010", "b1100011", "o60", "d48", "o65", "b1100001", "x63", "b1101
10",
     "d101", "o63", "b111001", "d97", "d51", "o70", "d55", "b1100010", "d125", "x20", "b101110", "x20", "b100100
0",
     "d97", "d118", "o145", "x20", "d97", "o40", "d103", "d111", "d111", "x64", "d32", "o164", "b1101001", "x6d"
,
     "o145", "x7e"]
s1 = ""
t = ""
t1 = ""
for i in s:
    s1 = i
    for j in range(1):
        if s1[0:1] == 'd':
            t = str(hex(int(s1[1:])))
            t = t[2:]
            t1 = t1 + t
        if s1[0:1] == 'x':
            t = s1[1:]
            t1 = t1 + t
        if s1[0:1] == 'o':
            t = str(hex(int(s1[1:], 8)))
            t = t[2:]
            t1 = t1 + t
        if s1[0:1] == 'b':
            t = str(hex(int(s1[1:], 2)))
            t = t[2:]
            t1 = t1 + t
print type(t1), t1
print binascii.unhexlify(t1)
```

flag{1e4bf81a6394de5abc005ac6e39a387b}

affine

```
flag = "szzyfimhyzd"
flaglist = []
for i in flag:
    flaglist.append(ord(i)-97)
flags = ""
for i in flaglist:
    for j in range(0,26):
        c = (17 * j - 8) % 26
        if(c == i):
            flags += chr(j+97)
print(flags)
```

flag{affineshift}

Crack it

john shadow

flag{hellokitty}

rsa

e特别大，wiener attack

第一种方法：RsaCtfTool：

```
python3 RsaCtfTool.py --createpub -n 460657813884289609896372056585544172485318117026246263899744329237492701820
627219556007788200590119136173895989001382151536006853823326382892363143604314518686388786002989248800814861248 5
950753262770996453386949770974591685308987760072936957281019760694239716965242377552271870614182028499114791247 9
3990722597 -e 3546111024413075720565721818279258991983453502287537309310893932754639165444566268942454150961078 3
446577840953237318712531855461472259930179152891621283936812106603554100880826153450058602365276771227162578520 4
280964688004680328300124849680477105302519377370092578107827116821391826210972320377614967547827619 > test.pem
python3 RsaCtfTool.py --publickey test.pem --private > test.key
python3 RsaCtfTool.py --key test.key --dumpkey
```

解出pq的值

```
from libnum import n2s, s2n
import base64


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


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 = 35461110244130757205657218182792589919834535022875373093108939327546391654445662689424541509610783446577
8409532373187125318554614722599301791528916212839368121066035541008808261534500586023652767712271625785204280964
6880046803283001248496804771053025193773700925781078271168213918262109723203776149675478276 19
    # tmp = base64.b64decode("qzogS7X8M3ZOpkUhJJcbukaRduLyqHAPblmabaYSm9iatuulrHcEpBmil7V40N7gbsQXwYx5EBH5r5V2HR
cEIOXjgfk5vpGLjPVxBLyXh2DajHPX6KvbFpQ8jNpCQbUNq8Hst00yDSO/6ri9dk6bk7+uyuN0b2K1bNG5St6sCQ4qYEA3xJbsHFvMqtvUdhMiqO
7tNCUVTKZdN7iFvSJqK2IHosIf7FqO24zkHZpHi31sYU7pcgYEaGkVaKs8pjq6nbnffr4URfoexZHeQtq5UAkr95zD6WgvGcxaTDKafFntboX9GR
9VUZnHePiio7nJ3msfue5rkIbISjmGCAlj+w==")
    d = modinv(e, (p - 1) * (q - 1))
    # c=s2n(tmp)
    c = 38230991316229399651823567590692301060044620412191737764632384680546256228451518238842965221394711848337
8324594438444468894683621541882148407367446578858589438101776758719911114666531582571911396056999163473082949956
6453028081685048274053060225455912375912110633835922024263777591902693356332606944942439 1192
    n = p * q
    m = pow(c, d, n)
    print (n2s(m))
```

flag{Wien3r_4tt@ck_1s_3AsY}

第二种方法：wiener attack 中直接求出d

```
import gmpy2
from libnum import n2s

n = '46065781388428960989637205658554417248531811702624626389974432923749270182062721955600778820059011913617389
59890013821515360068538233263828923631436043145186863887860029892488008148612485950753262770996453386949770974591
68530898776007293695728101976069423971696524237755227187061418202849911479124793990722597'

e = '35461110244130757205657218182792589919834535022875373093108939327546391654445662689424541509610783446577840
95323731871253185546147225993017915289162128393681210660355410088082615345005860236527677122716257852042809646880
04680328300124849680477105302519377370092578107827116821391826210972320377614967547827619'

enc = '38230991316229399651823567590692301060044620412191737764632384680546256228451518238842965221394711848337832
45944384444688946836215418821484073674465788585894381017767587199111146665315825719113960569991634730829499566
45302808168504827405306022545591237591211063383592202426377759190269335633260694494243911192'

p = '15991846970993213322072626901560749932686325766403404864023341810735319249066370916090640926219079368845510
44403140032222914777168296113242048189736284319 9'

q = '28805791771260259486856902729020438686670354441296247148207862836064657849735343618207098163901787287368569
76847252134463556733429935676008050745464020700 3'

d = '82646679722942750172933397723717833221688221494719768342210823934093636918 95'

n1 = gmpy2.mpz(n)
enc1 = gmpy2.mpz(enc)
d1 = gmpy2.mpz(d)

r = gmpy2.powmod(enc1, d1, n1)
print (r)
s = n2s(r)
print (s)
```

来自宇宙的信号

标准银河字母(Standard Galactic Alphabet) = > NOPQRST

flag{nopqrst}