# NCTF Crypto WriteUp

原创

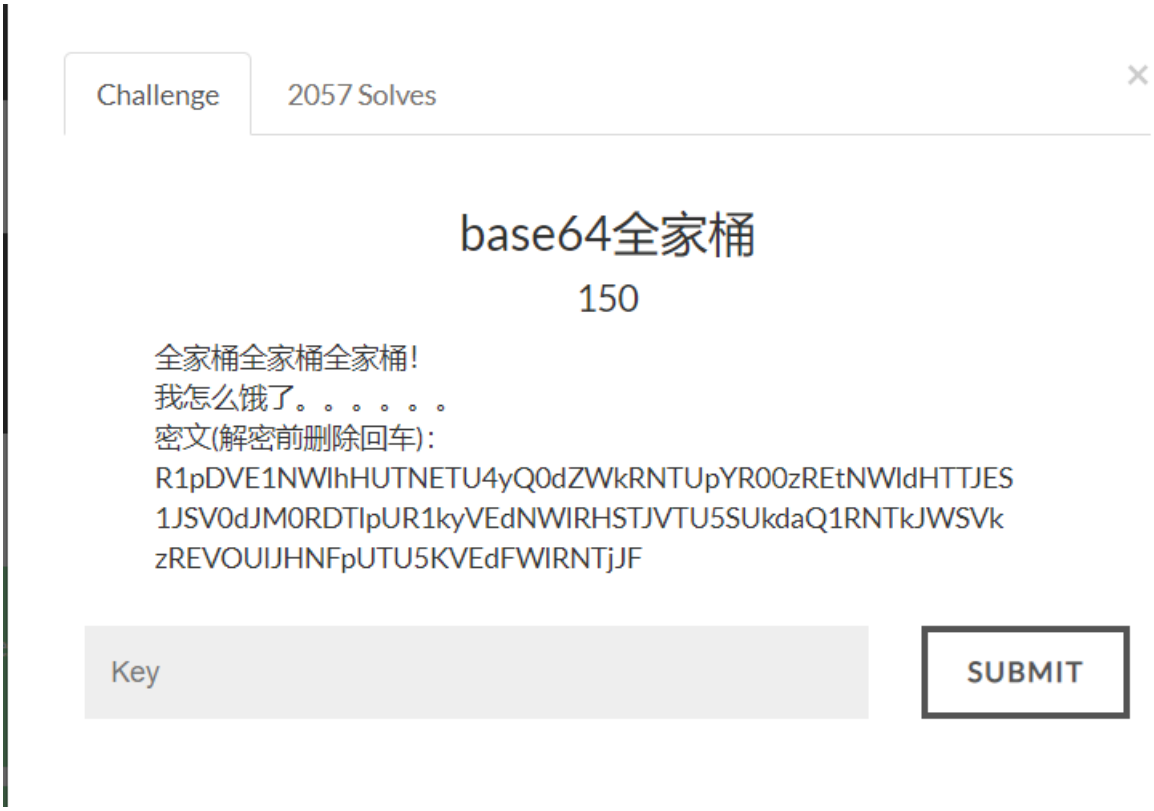
旗木家的卡卡西 于 2019-01-01 14:47:16 发布 371 收藏

http://ctf.nuptzj.cn/challenges

Crypto篇：

第一题、第二题、第七题和CG-CTF一样，不写了…

第三题：




说了全家桶，那就python跑吧…

```
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:57:15) [MSC v.1915 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import base64
>>> print(base64.b16decode(base64.b32decode(base64.b64decode('R1pDVE1NWlhHUTNETU4yQ0dZWkRNTUpYR00zREtNWldHTTJES1JSV0dJM0
RDTlpUR1kyVEdNWlRHSTJVTU5SUkdaQ1RNTkJWSVkzREVOUlJHNFpUTU5KVEdFWlRNTjJF'))))
b'nctf{base64_base32_and_base16}'
>>>
```

Flag：nctf{base64_base32_and_base16}

150分到手

第四题：


鬼知道进行了几次…

首先把base.txt中的回车都搞了，Notepad++吧，嗯。\r\n替换为空搞定。

搞完了就写个脚本跑一下，先跑他个100次试试…

```
# coding: utf-8

import base64

f = open('base64.txt','r')

str = f.read()

for i in range(0,100):

       str = base64.b64decode(str)

       print (str)
```

跑到出错，flag就出现了

```
b'V20xNGFGcDZjSFZaTTFKdFpUTkNjMXBYUm5wYVZqa3hZekpXWm1OSWJEQmhSemwxV0ROU2RsZ31VbXhaTWpsc1dsWTVhVmxZVG14T2FsSTU='
b'Wm14aFp6cHVZM1JtZTNCc1pXRnpaVjkxYzJWZmNIbDBhRz11WDNSd1gyUmxZMj1rW1Y5aV1YTmx0a1I5'
b'ZmxhZzpuY3Rme3BsZWFzZV91c2VfcH10aG9uX3RvX2R1Y29kZV9iYXN1NjR9'
b'flag:nctf{please_use_python_to_decode_base64}'
Traceback (most recent call last):
  File "9.py", line 26, in <module>
    str = base64.b64decode(str)
  File "C:\Python\Python37\lib\base64.py", line 87, in b64decode
    return binascii.a2b_base64(s)
binascii.Error: Invalid base64-encoded string: number of data characters (37) cannot be 1 more than a multiple of 4
```

Flag：nctf{please_use_python_to_decode_base64}

200分到手。

用Python解base64，嗯，记住了。

第五题：

Challenge 1004 Solves

# 骚年来一发吗

250

密文：iEJqak3pjIaZ0NzLiITLwWTqzqGAtW2oyOTq1A3pzqas

给了个php函数。

```
function encode($str){
        $_o = strrev($str);
        for($_0=0;$_0<strlen($_o);$_0++){
                $_c = substr($_o,$_0,1);
                $__ = ord($_c)+1;
                $_c = chr($__);
                $_= $_.$_c;
        }
        return str_rot13(strrev(base64_encode($_)));
}
```

大概思路就是先反转字符串，然后每一位都把它的ASCII码加一，然后反向打印base64，再rot13编码…

咋这么多步骤，那就php走起吧…

写了个php

```
<?php

$str = "iEJqak3pjIaZ0NzLiITLwWTqzqGAtW2oyOTq1A3pzqas";

$str = base64_decode(strrev(str_rot13($str)));

$_str = "";

for ($i=0;$i<strlen($str);$i++) {

       $c = substr($str,$i,1);

       $__ = ord($c) - 1;

       $c = chr($__);

       $_str = $_str.$c;

}

echo strrev($_str);

?>
```

放到phpStudy中跑一下，出结果。

Flag：nctf{rot13_and_base64_and_strrev}

250分到手

第六题：


给了个py

```
import random
from base64 import *
result={
    '16':lambda x:b16encode(x),
    '32':lambda x:b32encode(x),
    '64':lambda x:b64encode(x),
    }

flag=b"nctf{**********}"
for i in range(10):
    a=random.choice(['16','32','64'])
    flag=result[a](flag)

with open("code.txt","wb") as f:
    f.write(flag)
```

woc你是有毒吗？

人工解密吧，反正就十次…

第一次一看没有小写字母，base32，第二次只有数字，应该是base16，第三次只有数字和A-F字母，base16，第四次大小写都有，base64，第五次，base16，第六次，base64，第七次，base32，第八次，base16，第九次，base32，第十次，base32，然后出来了

nctf{random_mixed_base64_encode}

真累，有没有别的办法…

百度了一下别的wp，

https://www.jianshu.com/p/b1e1214a72cb

上面那个师傅是按照爆破做的，学习了。

Flag：nctf{random_mixed_base64_encode}

第八题：


Python大法好

```
# coding: utf-8

import hashlib

str1 = 'TASC'

str2 = 'O3RJMV'

str3 = 'WDJKX'

str4 = 'ZM'

for i in range(ord('A'),ord('Z') + 1):

       for j in range(ord('A'),ord('Z') + 1):

              for k in range(ord('A'),ord('Z') + 1):

                     str = str1 + chr(i) + str2 + chr(j) + str3 + chr(k) + str4

                     md5str = hashlib.md5(str.encode("utf-8")).hexdigest()

                     print (str + ' ' + md5str + '\n')

                     if (md5str[0:5]=='e9032'):

                            exit()
```

然后就成了

```
TASCJO3RJMVKWDJKXLZM e9032994dabac08080091151380478a2
```

其实应该再跑一次数字的，不过字母出了结果，数字就没必要跑了。

Flag：nctf{e9032994dabac08080091151380478a2}

300分到手

第九题：


加密算法看了一下，大概就是：

从一个文件中读取，如果读取不到换行符，就和给定的一个字符数组中的某个数异或，然后再以十六进制输出到另一个文件中。

嗯，好难...

首先我想的就是一个一个试吧...

```
#define KEY_LENGTH 2 // Can be anything from 1 to 13
```

这个不确定就一个一个试，这个

```
unsigned char key[KEY_LENGTH] = {0x00, 0x00};
/* of course, I did not use the all-0s key to encrypt */
```

明文首先是可见的字符吧...所以说可以排除掉几个

上C++，C++大法好...

```
#define _CRT_SECURE_NO_WARNINGS

#include <iostream>

#include <string>

using namespace std;

const int KEY_LENGTH_MAX = 13;

const int KEY_LENGTH_MIN = 1;

int main() {

    FILE *fpIn;

    unsigned char ch;

    string str = "\x00";

    fpIn = fopen("code.txt", "r");

    while (fscanf(fpIn, "%02X", &ch) != EOF) str += ch;

    fclose(fpIn);

    unsigned char ustr[471] = {};

    unsigned char xstr[471] = {};

    int a, b, c, d, e, f, g, h, k, l, m, n, p, i;

    for (i = 0; i < 471; i++) ustr[i] = str[i];

    unsigned char _key[13] = {};

    int KEY_LENGTH = KEY_LENGTH_MAX;

    for (; KEY_LENGTH != 0; KEY_LENGTH--) {

         for (a = 0; a < 256; a++) {

             _key[0] = a;

             for (b = 0; b < 256; b++) {

                  _key[1] = b;

                  for (c = 0; c < 256; c++) {
```

```
                    _key[2] = c;
                    for (d = 0; d < 256; d++) {
                        _key[3] = d;
                        for (e = 0; e < 256; e++) {
                             _key[4] = e;
                             for (f = 0; f < 256; f++) {
                                 _key[5] = f;
                                 for (g = 0; g < 256; g++) {
                                     _key[6] = g;
                                     for (h = 0; h < 256; h++) {
                                          _key[7] = h;
                                          for (k = 0; k < 256; k++) {
                                              _key[8] = k;
                                              for (l = 0; l < 256; l++) {
                                                   _key[9] = l;
                                                   for (m = 0; m < 256; m++) {
                                                       _key[10] = m;
                                                       for (n = 0; n < 256; n++) {
                                                           _key[11] = n;
                                                           for (p = 0; p < 256; p++) {
                                                                _key[12] = p;
                                                                unsigned char key[13] = {};
                                                                for (i = 0; i < 13; i++) key[i] = _key[12
                                                                for (i = 0; i < 470; i++) xstr[i] = key[i
                                                                cout << "当KEY_LENGTH=" << KEY_LENGTH << "
                                                                for (i = 0; i < KEY_LENGTH; i++) printf("
                                                                cout << "时，字符串解密如下：" << endl;
                                                                for (i = 0; i < 470; i++) if (xstr[i] >=
                                                                cout << endl;
                                                           }
```

```
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    return 0;
}
```

这个算法应该理论上是可行的，但是...按照这个算法跑估计跑到我结婚都跑不出来...

自闭了，还是看百度吧...

https://blog.csdn.net/jakekong/article/details/79884365

这篇文章有详细的解答

---

真的能跑出来！！！

```
当KEY_LENGTH=7，密码表的遍历为186 31 145 178 83 205 62 时，字符串解密如下：
Cryptography is the practice and study of techniques for, among other things, secure communication in the presence of at
tackers. Cryptography has been used for hundreds, if not thousands, of years, but traditional cryptosystems were designe
d and evaluated in a fairly ad hoc manner. For example, the Vigenere encryption scheme was thought to be secure for deca
des after it was invented, but we now know, and this exercise demonstrates, that it can be broken very easily.
```

---

学习中...