异性相吸_writeup

原创

o_wow 于 2016-05-19 00:22:18 发布 1719 收藏

分类专栏： CTF 文章标签： ctf

本文链接：https://blog.csdn.net/o_wow/article/details/51449591

http://ctf.nuptzj.cn/challenges#异性相吸

vi +_+-.txt / +?+-.txt vim编辑器底部出现一排红色提示

Command can not be … 提示没要看仔细，真的麻烦很多 555

不识别+_+- -> 改文件名字

密码题一般套路

1、cat/strings/ghex/vi [-b(二进制打开) :%!xxd 16进制查看

2、base64 **循环多次解密**

hex -> dec ->ascii

```
base64.b16/32/64decode(a)   encode
```

3、md5 **32位 16进制**

4、xor

python 中进制间转换函数

异性相吸–by xq17

2016-02-13来源：xq17查看：117顶（0)踩(0)

哈哈，我是xq17 最简单的一道题看没人·来 小菜就来秀数学功底了

异或加密假设密钥为k 明文为A 密文为B

B = （A^K）

根据异或算法可逆嘛二进制运算01=0 11=1

A = B ^ K

A^B =A^(A^K)=K

所以 写个脚本来运算下落

```
# lusr/bin/env python
#  _*_ coding:utf-8 _*_
# author:xq17
a = open('input1.txt','rb').read()
b = open('input2.txt','rb').read()
oxstr =''
for i in range(0,len(a)):
    res = ord(list(a)[i]) ^ ord(list(b)[i])
    oxstr =oxstr +chr(res)
print oxstr
```

OK:flag:nctf{**}

`a = open('+_+-.txt','rb').read()` 时并未按16进制输出，而是

`'\n\x03\x17\x02V\x01\x15\x11\n\x14\x0e\n\x1e0\x0e\n\x1e0\x0e\n\x1e0\x14\x0c\x19\r\x1f\x10\x0e\x06\x03\x18'`

？还是不懂

但是 a[1] = '\x03' /'/n'/'/r' 一组一组

16进制不区分大小写（均可）

> `Pattern p = Pattern.compile( "[\\x00-\\x09\\x0b\\x0c\\x0e-\\x19]`
> "); 这里面的 `\\x00-\\x09\\x0b\\x0c\\x0e-\\x19` 意思
> 有些Unicode字符不允许出现在URI引用（URI Reference）中。因此，处理器必须对这些字符进行编码和转义处理，以得到一个合法的URI引用。
> 不允许在URI引用中出现的字符包括：所有的非ASCII字符，以及在[IETF RFC 2396]2.4节列出的字符中去掉"#"、"%"和方括号符号后剩下的符号。必须按照下面的方式对这些非法字符进行转义处理：
>
> 将每个非法字符转换为UTF-8格式[IETF RFC 2279]（UTF-8用一个或多个字节来表示原来的字符）。
>
> 对于每个与非法字符相应的字节[每个非法字符转换为UTF-8格式后，有一个或多个与它相应的字节]，按照URI转义操作机制对它进行转义（也就是说，转换为%HH形式，这里HH是字节值的十六进制记法）。
>
> 用最终的字符序列来替换原来的字符。
> http://www.educity.cn/wenda/434285.html

BinHex 乱码

> BinHex 编码是 Macintosh 计算机上用可打印字符表示/传输二进制文件的一种编码方法。目前通用的 BinHex 4.0 的这种编码的文件一般以 .HQX 为后缀名。早期的 BinHex2.0 编码文件一般以 .HEX 为文件名后缀。 BinHex 4.0 是一种带有 CRC 校验的编码。在一些 email 程序中 (如使用最广泛的邮件程序之一 Eudora)，BinHex 编码是用于 Attach 二进制文件的方法之一。
>
> 但是有很多广泛使用的 email 程序不支持这种格式，如Microsoft OutLook Express 接收 Eudora 发来的以 BinHex 夹带的二进制文件时，只能分辨出夹带的文件，却不能正确解码。类似的情况需要将信件 Forward 到一个可以解析相应格式的邮件程序中或对邮件原文进行人工处理。
>
> BinHex 编码是这个样子的：

```
(This file must be converted with BinHex 4.0)
 :#dC*@8"A5812@MP3!("D59"`@MP3!!!!!%$)!!!!!'eC8%X$""3!!J!)!0e44b1
 NrPJL%N!!!!"d!!!,!!!!4NPB8&G*6Lj&@%AX[AeJ8dA@1$c*6@l5I0`@U18lK)p
 ')DK8#Y3MHLKmPJ+B8T&LJ3"&D6*0@A#KK5p$MP[UeUl$2I5$52Z[(ZL"&!LL ......
 !!!!!!3!)+bfS!!!!"'59K39dP1,N9B43F!1J"dH(#V#,jTFELAU2ql`1-5eAr
 iVS(5RqX,rF9@h&M(%R)a@8flJFd'0dpD@i$pVJ"FFBTf'@a3V1Sb8%X&"J!!!!
 "!!%!G`!!!$Y!!!!!"+D!!!:
```

它的开始行必定是“(This file must be converted with BinHex 4.0)”，整个数据块以冒号开始、并以冒号结尾。使用 BinHex 编码的邮件一般应该在信头中含有类似下面这样的说明(假设Attach文件名为filename.ext的话)：

```
Content-Type: application/mac-binhex40; name="filename.ext"
Content-Disposition: attachment; filename="filename.ext"
```

将含有数据块的文件更名为 .HQX，即可双击该文件启动 Winzip 进行解码。(http://www.winzp.com)。至此，我们不得不赞叹 winzip 在解码工作中的无以伦比的表现 (其支持的后缀名有：*.zip*、 .z、 *.gz*、.tz、*.taz*、.tgz、*.lzh*、.arj、*.arc*、.tar、.exe(ziped)、 *.uu*、 .uue、 *.xxe*、.bhx、 *.b64*、.hqx)。遗憾的是 Winzip 对 BinHex 的解码并不总是成功的。在测试某一封 Eudora 发出的 BinHex 编码信件的时候，Winzip 不能解码。

一些软件支持BinHex解码，它们同时大多还支持一些其他编码。如 StuffIt Expander (ftp://ftp.aladdinsys.com/ 或找其他共享软件网点)、 Fastcode32 (http://www.freewarehome.com/utilities/encrypt.html) 等，一些网点 (如 http://helpdesk.uvic.ca/how-to/support/unix/hqx/unhqx.c)还提供了BinHex 解码的源程序。

BinHex 4.0 是由 Yves Lempereur 在 84-85 年开发的，这是目前最通用的版本，在 Mac/Unix/PC 上广泛运用。Yves 还开发了一个与 MacBinary (Mac上 面的另一种编码) 兼容的 BinHex 5.0 版本。 BinHex 5.0 与 BinHex 4.0 不兼容，它们是两种截然不同的编码。 BinHex 比一般编码耗费更多的字节，并且跨平台的解码工具比其他编码少。http://www.5dmail.net/html/2006-3-2/200632222823.htm

python unicode转str方法

```
import unicodedata
s = u"Marek Čech"    #(u表示是unicode而非 ascii码，不加报错！)
line = unicodedata.normalize('NFKD',s).encode('ascii','ignore')
print line

>>> Marek Cech
```

python _ Unicode

关于Unicode的详细情况可以参考百度百科：http://baike.baidu.com/view/40801.htm
这里简单的说一下。（下面内容基本上时从《Python.Core.Programming.2ed》上摘的）
Unicode是计算机可以支持这个星球上的多种语言的秘密武器，在Unicode之前，用的都是ASCII，ASCII吗非常简单，每个英文字符都用7位二进制数的方式存储在计算机内，其范围是32到126.它的实现原理这里也不说了。
但是ASCII码只能表示95个可打印的字符，后来把ASCII扩展到了8位，这样就能表示223个字符了，虽然这个来表示欧美字母语言已经足够了，但是对于像中文等语系来说就太少了。于是Unicode码诞生了。
Unicode通过使用一个或者多个字节来表示一个字符，这样就突破了ASCII的限制，这样，Unicode可以表示超过90000个字符了。
Python 与Unicode
为了让Unicode和ASCII码值的字符串看起来尽可能的相像，Python的字符串从原来的简单数据类型改变成了真正的对象，ASCII字符串成了ＳｔｒｉｎｇＴｙｐｅ，而Ｕｎｉｃｏｄｅ字符串成了UnicodeType类型，他们的行为非常相近。String模块里面都有相应的处理函数。String模块已经停止了更新，只保留了对ＡＳＸＩＩ码的支持，string模块已经不推荐使用，在任何要跟Ｕｎｉｃｏｄｅ兼容的代码里都不要再用该模块，Python保留该模块仅仅为了向后兼容。
Python里面默认所有字面上的字符串都用ASCII编码，可以通过在字符串前面加一个‘ｕ’前缀的方式声明Ｕｎｉｃｏｄｅ字符串，这个‘ｕ’前缀告诉Python后面的字符串要编成Ｕｎｉｃｏｄｅ字符串。

```
>>> "Hello World" #ASCII string
'Hello World'
>>> u"Hello World" #Unicode string
u'Hello World'
```

内建的str()函数和chr（）函数不能处理Unicode，它们只能处理常规ASCII编码的字符串，如果一个Unicode字符串作为参数传给了str（）函数，它会首先被转换成ASCII码字符串然后交给str（）函数。
Codecs
Codec是把Coder/DECoder得首字母组合，它定义了文本跟二进制的转换方式，跟ASCII那种用一个字节把字符转换成数字的方式不同，Unicode用的是多字节，这导致了Unicode支持多种不同的编码方式，比如说codec支持的四种耳熟能详的编码方式是：
ASCII，ISO8859—1/Latin-1，UTF-8,和UTF-16
最著名的是UTF-8编码，它也用一个字节来编码ASCII字符，这让那些必须同时处理ASCII码和Unicode码文本的程序员的工作变得非常轻松，因为ASCII字符的UTF-8编码和ASCII编码完全相同。
UTF-8编码可以用1到4个字节来表示其他语言的字符，这给那些需要直接处理Unicode数据的程序员带来了麻烦，因为他们没有办法按照固定长度逐一读出各个字符，幸运的是我们不需要掌握直接读取Unicode数据的方法，Python已经替我们完成了相关细节，我们无需为处理多字节字符的复杂问题而担心。
UTF-16也是一种变长编码，但是它不常用。
编码解码
Unicode支持多种编码格式，这为程序员带来了额外的负担，每当你向一个文件写入字符串的时候，你必须定义一个编码用于把对应的Unicode内容转换成你定义的格式，Python通过Unicode字符串的encode()函数解决了这个问题，该函数接受字符串中的字符为参数，输出你指定的编码格式的内容。
所以，每次我们写一个Unicode字符串到磁盘上我们都要用指定的编码器给他“编码“一下，相应地，当我们从这个文件读取数据时，我们必须”解码”该文件，使之成为Unicode字符串对象。
简单的例子：
下面的代码创建了一个Unicode字符串，用UTF-8编码器将它编码，然后写入到一个文件中去，接着把数据从文件中读回来，解码成Unicode字符串对象，最后，打印出Unicode字符串，用以确认程序正确地运行。
在Linux中编写，在VIM中输入如下代码，保存为uniFile.py，红字是我加的注释

```
An example of reading and writing Unicode strings:Writes

a Unicode string to a file in utf-8 and reads it back in

'''

CODEC = 'utf-8'          编码方式

FILE = 'unicode.txt'     要存的文件名



hello_out = u"Hello world\n"    创建了一个Unicode格式的字符串

bytes_out = hello_out.encode(CODEC)    用UTF-8编码

f = open(FILE,'w')

f.write(bytes_out)              写入指定文件中

f.close()



f = open(FILE,'r')

bytes_in = f.read()           读取

f.close()

hello_in = bytes_in.decode(CODEC)     解码

print hello_in                        打印
```

结果打印出 Hello world
然后我们在python目录下会发现多了一个名为unicode.txt的文件，用cat命令查看一下，发现里面的内容和打印的结果一样.
把Unicode应用到实际中注意一下四点：
1 程序中出现字符串时一定要加一个前缀u
2 不要用str（）函数，用Unicode（）代替
3 不要用过时的string模块。如果传给它非ASCII码，它会把一切搞砸。
4 不到必须时不要在你的程序里编解码Unicode字符，只在你要写入文件或者数据库或者网络时，才调用encode（）函数和decode（）函数。
http://www.cnblogs.com/jackge/archive/2013/06/04/3117352.html