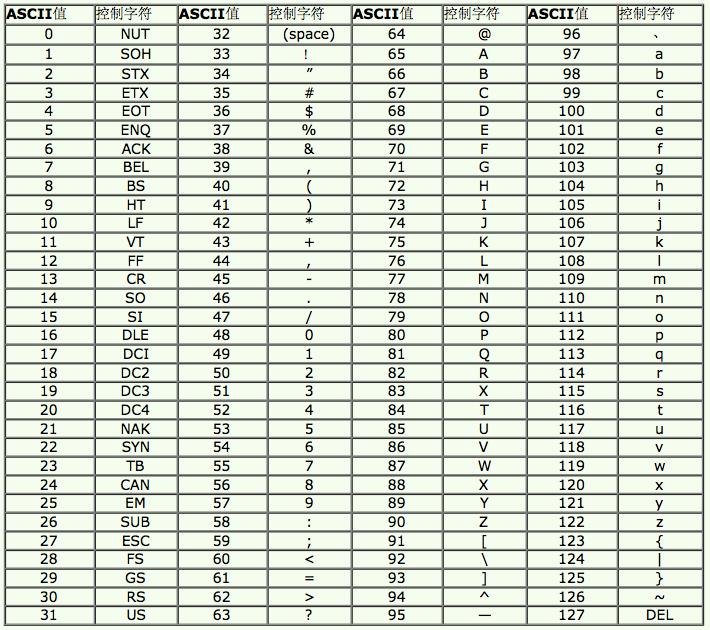
有关于MISC中编码及音频隐写的相关总结

隐写术：

将信息隐藏到信息载体，不让计划的接收者之外的人获取信息。近几年来，隐写术领域已经成为了信息安全的焦点。因为每个Web站点都依赖多媒体,如音频、视频和图像。隐写术这项技术可以将秘密信息嵌入到数字媒介中而不损坏它的载体的质量。第三方既觉察不到秘密信息的存在,也不知道存在秘密信息。因此密钥、数字签名和私密信息都可以在开放的环境(如Internet或者内联网)中安全的传送

**一、编码隐写：**

在misc中有很多地方使用了密码学的原理，其中较常见的有base编码、猪圈、栅栏、摩斯等等，关于密码学的总结之后再说，鉴于编码隐写在杂项中往往不会单独考察，一般都是和音频或图片隐写结合起来，成为题目的一部分，因此本文所选的题目钧较为简单，主要考察某个特定编码的解读，大多数编码都可以使用在线网站或python脚本进行解码，因此这里仅总结在misc中常用到的几类编码和本人遇见过的一些特殊编码，由于编码种类过多，因此只能一步步的积累，在做题时经常会遇到一些稀奇古怪的编码类型。

第一个介绍的最基础的编码莫过于ASCII编码，ASCII编码相信是大部分人最熟悉的编码之一，它是西欧编码，仅仅用一个字节表示字符，ASCII码使用指定的7位或8位二进制数组合来表示128或256种可能的字符。标准ASCII码也叫基础ASCII码，使用7位一进制数(剩下的1位一进制为0)来表示所有的大写和小写字母，数字0到9、标点符号，以及在美式英语中使用的特殊控制字符。前128个ASCII(0~127)为标准ASCII字条,第7位为0,高位为1的128个ASCII是扩展ASCII符.(我们平常使用的基本都在标准字符内.)

那么在杂项中我们在附件中或者当你把某个文件放在010/winhex中，有时会出现一串ascii编码，这个时候我们要认出这种编码，一般不需要我们自己去算，网上有很多在线的工具平台，可以自行搜索，比如在bugku自带的工具库就可以进行解码。

此外，ascii码通常不会单独拿出来考，经常会做一些变形，如二进制编码 将 ascii 码对应的数字换成二进制表示形式。特点为只有 0 和 1、不大于 8 位，一般 7 位也可以，因为可见字符到 127。这其实是另一种 ascii 编码。

还有十六进制编码，将 ascii 码对应的数字换成十六进制表示形式。A-Z→41-5 A，

1. z→61-7 A

工具附上：jpk, ascii to number, number to ascii <http://www.ab126.com/goju/1711.html>

### 题目参考：2018 DEFCON Quals ghettohackers: Throwback

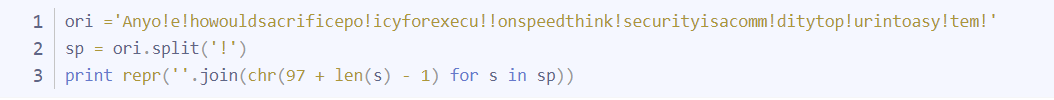
题目如下：



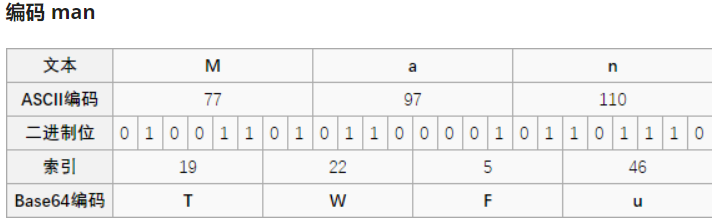
这里首先考虑是否需要补全！的位置，试着补全之后发现并不符合flag的格式，所以考虑！是用来分割字符的，我们将字符串长度1对应字母a，长度2对应字母b，以此类推，同时要注意长度0对应空格，归类后发现flag

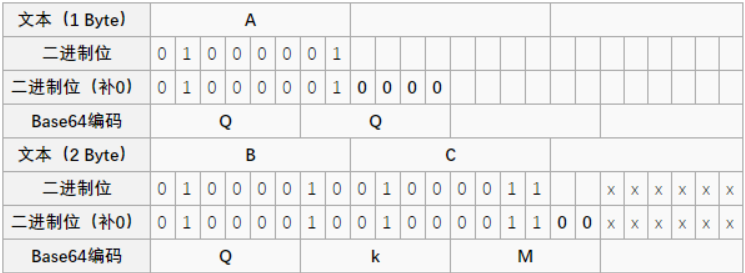


也可以使用脚本进行分析，效率会更快一些

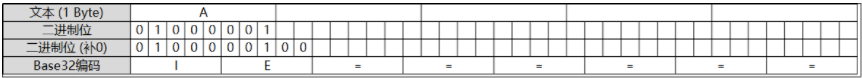


第二种编码是base编码，这种编码经常出现在杂项中，往往隐藏在一大堆乱码当中，base xx 中的 xx 表示的是采用多少个字符进行编码，比如说 base64 就是采用以下 64 个字符编码，由于2的6次方等于 64，所以每 6个比特为一个单元，对应某个可打印字符。3个字节就有 24个比特，对应于4个Base64 单元，即 3个字节需要用4个可打印字符来表示。它可用来作为电子邮件的传输编码。在 Base64 中的可打印字符包括字母 A-Z、a-z、数字 0-9，这样共有 62个字符，此外两个可打印符号在不同的系统中而不同。



如果要编码的字节数不能被 3 整除，最后会多出 1 个或 2 个字节，那么可以使用下面的方法进行处理：先使用 0 值在末尾补足，使其能够被 3 整除，然后再进行 base64 的编码。在编码后的 base64 文本后加上一个或两个 = 号，代表补足的字节数。也就是说，当最后剩余一个八位字节（一个 byte）时，最后一个 6 位的 base64 字节块有四位是 0 值，最后附加上两个等号；如果最后剩余两个八位字节（2 个 byte）时，最后一个 6 位的 base 字节块有两位是 0 值，最后附加一个等号。由于解码时补位的 0 并不参与运算，可以在该处隐藏信息。

与 base64 类似，base32 使用 32 个可见字符进行编码，2 的 5 次方为 32，所以每 5 bit 为 1 个分组。5 字节为 40 bit，对应于 8 个 base32 分组，即 5 个字节用 8 个 base32 中字符来表示。但如果不足 5 个字节，则会先对第一个不足 5 bit 的分组用 0 补足了 5 bit ，对后面剩余分组全部使用 “=” 填充，直到补满 5 个字节。由此可知，base32 最多只有 6 个等号出现。



特点：base64 结尾可能会有 = 号，但最多有 2 个；base32 结尾可能会有 = 号，但最多有 6 个；根据 base 的不同，字符集会有所限制；有可能需要自己加等号。

Base家族种类众多，有的如base64、32、16等有显著的特点可以看出，有的则只能自己去进行尝试。

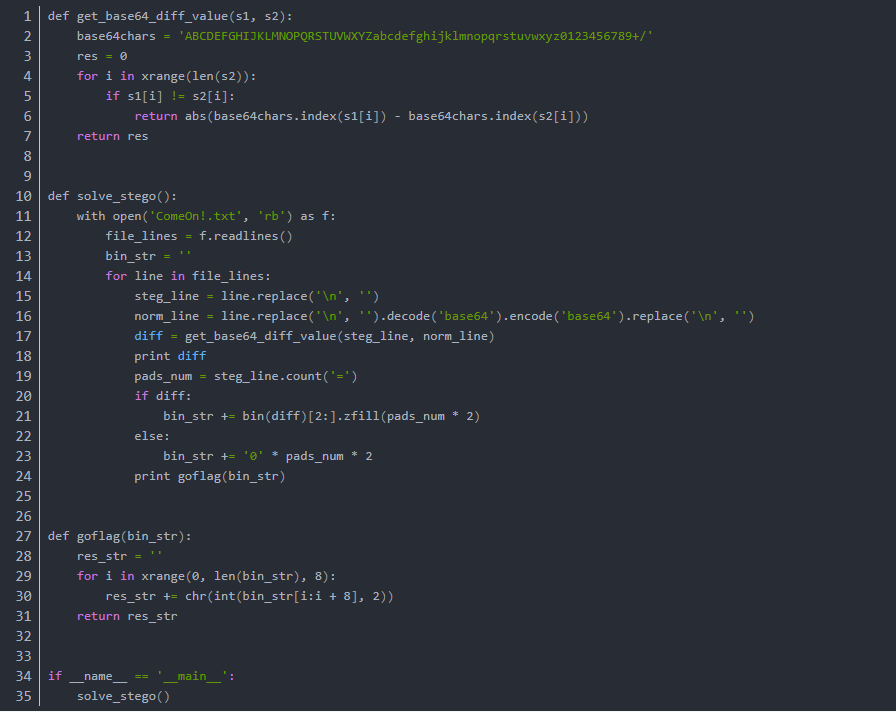
工具附上：http//www.qtool.net/baseencode可以在线解码包括base16，32，62，64，81等多种base编码。

 Basecrack，工具，可以自动识别是哪一种base

题目：

[ctf-challenges/misc/encode/computer/base64-stego at master · ctf-wiki/ctf-challenges (github.com)](https://github.com/ctf-wiki/ctf-challenges/tree/master/misc/encode/computer/base64-stego)。

见data.txt文件

解题可以直接复制到工具中，也可以使用脚本编译

得到flag{6aseb4\_f33!}

第三种介绍URL编码，这种编码以大量的百分号为特点，url编码是一种浏览器用来打包[表单](https://baike.baidu.com/item/%E8%A1%A8%E5%8D%95/5380322" \t "https://baike.baidu.com/item/URL%E7%BC%96%E7%A0%81/_blank)输入的格式。我们输入的网址实质上就是URL编码，URL 只能使用 [ASCII 字符集](https://www.runoob.com/tags/ref-ascii.html)来通过因特网进行发送。由于 URL 常常会包含 ASCII 集合之外的字符，因此URL 必须转换为有效的 ASCII 格式。URL 编码使用 "%" 其后跟随两位的十六进制数来替换非 ASCII 字符。URL不能包含空格。且URL 编码通常使用+来替换空格。

例如用URL转码汉字时要先将汉字转化为ascii编码，然后转为十六进制，再在前面加上%。本题未找到相应题目，读者在做题过程中只要能认出即可。

例：%61%6c%66%7b%67%79%72%63%74%70%72%67%6f%70%61%69%5f%79%5f%73%73%61%65%7d%79

工具附上：[www.bejson.com/enc/urlencode/](http://www.bejson.com/enc/urlencode/)

## IMG_256第四种介绍Morse Encoding（摩尔斯密码），莫尔斯密码由“.-”组成，最多有六位数字，也可以由01组成，摩尔斯密码运用十分广泛，且形式多样，在misc种可以用在音频，图片，像素，解码等，往往不会直接给出，要通过做题者将题目内容进行一定程度的转化，如音频的长短或黑白颜色，以及其他形式的转码，摩尔斯电码由两种基本信号组成：短促的点信号“·”，读“滴”；保持一定时间的长信号“—”，读“嗒”

工具附上：[摩斯电码转换\_摩斯密码翻译器-在线工具 (all-tool.cn)](http://www.all-tool.cn/Tools/morse/?&rand=d52dd921b1e3351395af1f4a268eec82)

题目附上：ctf.idf.cn/index.php?g=game&m=article&a=index&id=33

嘀嗒嘀嗒嘀嗒嘀嗒 时针它不停在转动

--  ---  .-.  ...  .

嘀嗒嘀嗒嘀嗒嘀嗒 小雨它拍打着水花

-.-.  ---  -..  .

PS：答案格式wctf{你所知道的}

参照上表得：wctf{morsecode}

第五种介绍HTML实体编码，当浏览器向 Web 服务器请求服务时，可能会发生错误，并且服务器可能会返回错误代码，例如 "404 Not Found"。通常这些错误被称为 HTML 错误消息。但是实际上这些消息应称为 HTTP 状态消息。实际上，服务器总会为每个请求返回一条消息。最常见的消息是 200 OK。超文本传输协议（Hypertext Transfer Protocol，缩写 HTTP）旨在启用客户端和服务器之间的通信。HTTP 充当客户端和服务器之间的请求-响应协议。

对于这种编码，我们可直接将码输入HTML中，然后直接打开

工具附上：www.runoob.com/runcode

第六种编码是常见的Unicode编码，这种编码将世界上所有的符号都纳入其中。每一个符号都给予一个独一无二的编码，那么乱码问题就会消失。这就是 Unicode，是一种所有符号的编码。Unicode是一个很大的集合，现在的规模可以容纳100多万个符号。每个符号的编码都不一样，比如，U+0639表示阿拉伯字母Ain，U+0041表示英语的大写字母A，U+4E25表示汉字严。具体的符号对应表，可以查询[unicode.org](https://www.unicode.org/" \t "https://www.ruanyifeng.com/blog/2007/10/_blank)，或者专门的[汉字对应表](http://www.chi2ko.com/tool/CJK.htm" \t "https://www.ruanyifeng.com/blog/2007/10/_blank)。

比如，汉字严的 Unicode 是十六进制数4E25，转换成二进制数足足有15位（100111000100101），也就是说，这个符号的表示至少需要2个字节。表示其他更大的符号，可能需要3个字节或者4个字节，甚至更多。

但是Unicode 只是一个符号集，它只规定了符号的二进制代码，却没有规定这个二进制代码应该如何存储。这里就有两个严重的问题，第一个问题是，如何才能区别 Unicode 和 ASCII ？计算机怎么知道三个字节表示一个符号，而不是分别表示三个符号呢？第二个问题是，我们已经知道，英文字母只用一个字节表示就够了，如果 Unicode统一规定，每个符号用三个或四个字节表示，那么每个英文字母前都必然有二到三个字节是0，这对于存储来说是极大的浪费，文本文件的大小会因此大出二三倍，这是无法接受的。

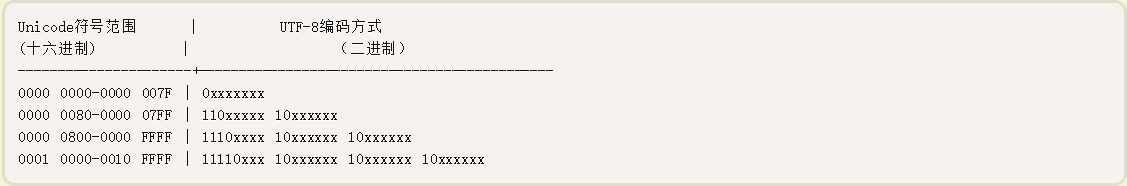
① UTF-8：

互联网的普及，强烈要求出现一种统一的编码方式。UTF-8 就是在互联网上使用最广的一种 Unicode 的实现方式。其他实现方式还包括 UTF-16（字符用两个字节或四个字节表示）和 UTF-32（字符用四个字节表示），不过在互联网上基本不用。重复一遍，这里的关系是，UTF-8 是 Unicode 的实现方式之一。UTF-8 最大的一个特点，就是它是一种变长的编码方式。它可以使用1~4个字节表示一个符号，根据不同的符号而变化字节长度。

Utf-8的规则简单，共两条。

1）对于单字节的符号，字节的第一位设为0，后面7位为这个符号的 Unicode 码。因此对于英语字母，UTF-8 编码和 ASCII 码是相同的。

2）对于n字节的符号（n > 1），第一个字节的前n位都设为1，第n + 1位设为0，后面字节的前两位一律设为10。剩下的没有提及的二进制位，全部为这个符号的 Unicode 码。

下表总结了编码规则，字母x表示可用编码的位。

跟据上表，解读 UTF-8 编码非常简单。如果一个字节的第一位是0，则这个字节单独就是一个字符；如果第一位是1，则连续有多少个1，就表示当前字符占用多少个字节。

下面，还是以汉字严为例，演示如何实现 UTF-8 编码。

严的 Unicode 是4E25（100111000100101），根据上表，可以发现4E25处在第三行的范围内（0000 0800 - 0000 FFFF），因此严的 UTF-8 编码需要三个字节，即格式是1110xxxx 10xxxxxx 10xxxxxx。然后，从严的最后一个二进制位开始，依次从后向前填入格式中的x，多出的位补0。这样就得到了，严的 UTF-8 编码是11100100 10111000 10100101，转换成十六进制就是E4B8A5。

② Little endian 和Big endian：

UCS-2 格式（即直接用两个字节存入字符的 Unicode 码）可以存储 Unicode 码（码点不超过0xFFFF）。以汉字严为例，Unicode 码是4E25，需要用两个字节存储，一个字节是4E，另一个字节是25。存储的时候，4E在前，25在后，这就是 Big endian 方式；25在前，4E在后，这是 Little endian 方式。第一个字节在前，就是"大头方式"（Big endian），第二个字节在前就是"小头方式"（Little endian）

Unicode 规范定义，每一个文件的最前面分别加入一个表示编码顺序的字符，这个字符的名字叫做"零宽度非换行空格"（zero width no-break space），用FEFF表示。这正好是两个字节，而且FF比FE大1

如果一个文本文件的头两个字节是FE FF，就表示该文件采用大头方式；如果头两个字节是FF FE，就表示该文件采用小头方式

实例展示：以汉字“严”为例，分别使用ANSI、UTF-8、Big endian、 Unicode四种方式储存。

（ANSI是默认的编码方式。对于英文文件是ASCII编码，对于简体中文文件是GB2312编码（只针对 Windows 简体中文版，如果是繁体中文版会采用 Big5 码））

1）ANSI：文件的编码就是两个字节D1 CF，这正是严的 GB2312 编码，这也暗示 GB2312 是采用大头方式存储的。

2）Unicode：编码是四个字节FF FE 25 4E，其中FF FE表明是小头方式存储，真正的编码是4E25。

3）Unicode big endian：编码是四个字节FE FF 4E 25，其中FE FF表明是大头方式存储。

4）UTF-8：编码是六个字节EF BB BF E4 B8 A5，前三个字节EF BB BF表示这是UTF-8编码，后三个E4B8A5就是严的具体编码，它的存储顺序与编码顺序是一致的。

在我们做misc或web题的时候，可能会看到前面是/u或、&#"或者"&#x"开头的字符，这三种都可以用来表示一串Unicode码，在js中这个叫Unicode转义字符, 和\n、\r同属于转义字符. 在其他语言中也有类似的, 可能还有其它变形的格式，unescape是用来处理%uXXXX这样格式的字符串,（unescape和escape编码是两种不同格式的书写方式，其本质都是Unicode编码） 将\uXXXX替换成%uXXXX后unescape就可以处理了.和解码中相对应,使用escape编码, 然后将%uXXXX替换为\uXXXX, 因为escape还可能把一些字符编码成%XX的格式, 所以这些字符还需要使用unescape还原回来.

第七种编码介绍我们生活中很常见的一种编码，条形码，条形码有时会隐藏在题目中，需要进行一定的长款变化，通过在线识别即可，在我们做题时是一种较为简答的编码形式。

条形码由宽度不等的多个黑条和空白组成，按照一定的编码规则排列，用以表达一组信息的图形标识符。

国际标准：

EAN-13 商品标准，13 位数字；

Code-39：39 字符；

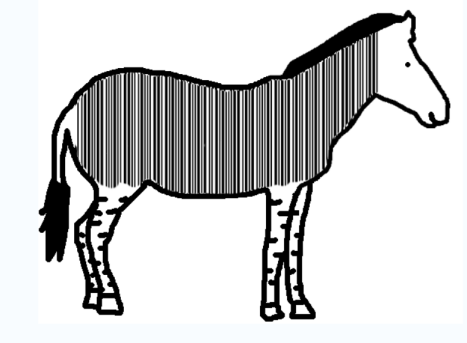
Code-128：128 字符

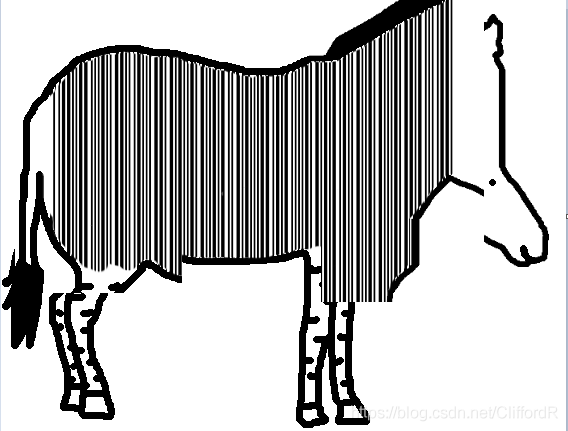
网站附上：online-barcode-reader.inliteresearch.com

也可以使用工具：bctester

题目附上：[题目 (xctf.org.cn)](https://adworld.xctf.org.cn/task/answer?type=misc&number=1&grade=1&id=4592&page=1)

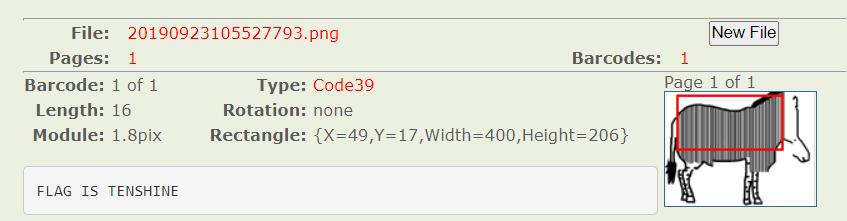
题目：Banmabanma

到手一个斑马的图片，观察发现斑马的身上是条形码。



为了便于扫描条形码，我们把条形码上下都拉长一点，这里有个小技巧，在画图软件中打开图片，使用虚线框住，然后按住alt+上下键就可以在上或者下方拉长图片。最后得到这么一个图片。

在线扫描得到flag



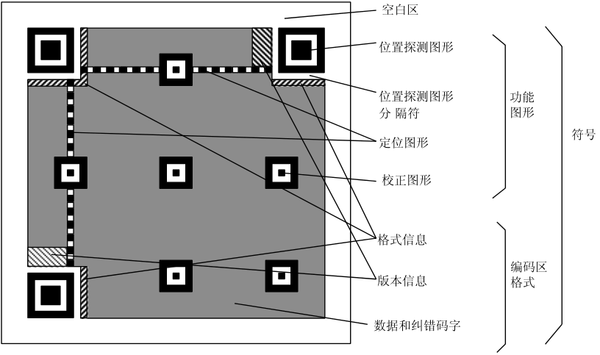
第八种编码我们介绍在Misc中最常见的二维码，二维码由于其特性，在Misc中考察极多，且考察形式多样复杂，一般和图片隐写相结合，通常隐藏在图片的某个通道里，或者最终解出来的图片二维码残缺需要读者自行补全，再或者与其他形式隐写结合，记得有一道题以01的二进制给出，诱导读者进行一系列转码操作，最终没什么用，实际要将0当作白色，1当作黑色，按一定顺序排列和涂抹，最终组成一个二维码。

二维条码是指在一维条码的基础上扩展出另一维具有可读性的条码，使用黑白矩形图案表示二进制数据，被设备扫描后可获取其中所包含的信息。一维条码的宽度记载着数据，而其长度没有记载数据。二维条码的长度、宽度均记载着数据。二维条码有一维条码没有的“定位点”和“容错机制”。容错机制在即使没有辨识到全部的条码、或是说条码有污损时，也可以正确地还原条码上的信息。二维条码的种类很多，不同的机构开发出的二维条码具有不同的结构以及编写、读取方法。

二维码包括几种不同的类型：

1、堆叠式 / 行排式二维码：Code 16 k、Code 49、PDF417

2、矩阵式二维码：QR CODE，二维码的名称是相对与一维码来说的，比如以前的条形码就是一个“一维码”。它的优点有：二维码存储的数据量更大；可以包含数字、字符，及中文文本等混合内容；有一定的容错性（在部分损坏以后可以正常读取）；空间利用率高等。



   位置探测图形、位置探测图形分隔符、定位图形：用于对二维码的定位，对每个QR码来说，位置都是固定存在的，只是大小规格会有所差异；

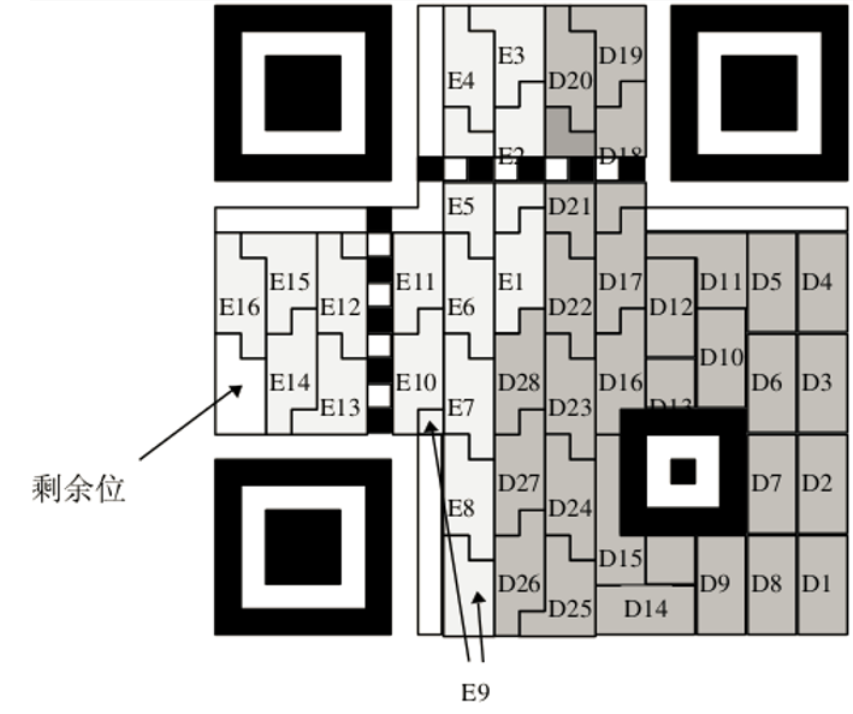
    校正图形：规格确定，校正图形的数量和位置也就确定了；

    格式信息：表示改二维码的纠错级别，分为L、M、Q、H；

     版本信息：即二维码的规格，QR码符号共有40种规格的矩阵（一般为黑白色），从21×21（版本1），到177×177（版本40），每一版本符号比前一版本 每边增加4个模块。

    数据和纠错码字：实际保存的二维码信息，和纠错码字（用于修正二维码损坏带来的错误）。

了解了二维码的基本结构后，将探测图形、分隔符、定位图形、校正图形和码字模块放入矩阵中，并把上面的完整序列填充到相应规格的二维码矩阵的区域中。



对于我们来说只需简单了解即可，不需要深究，在做题中不会考察我们对其原理的理解，而是更多的对于我们如何补全二维码，或如何正确探索出二维码其中的信息。

二维码最经典的补全当然是对定位角的补全。

1、首先随便找一个二维码，将定位角截下来

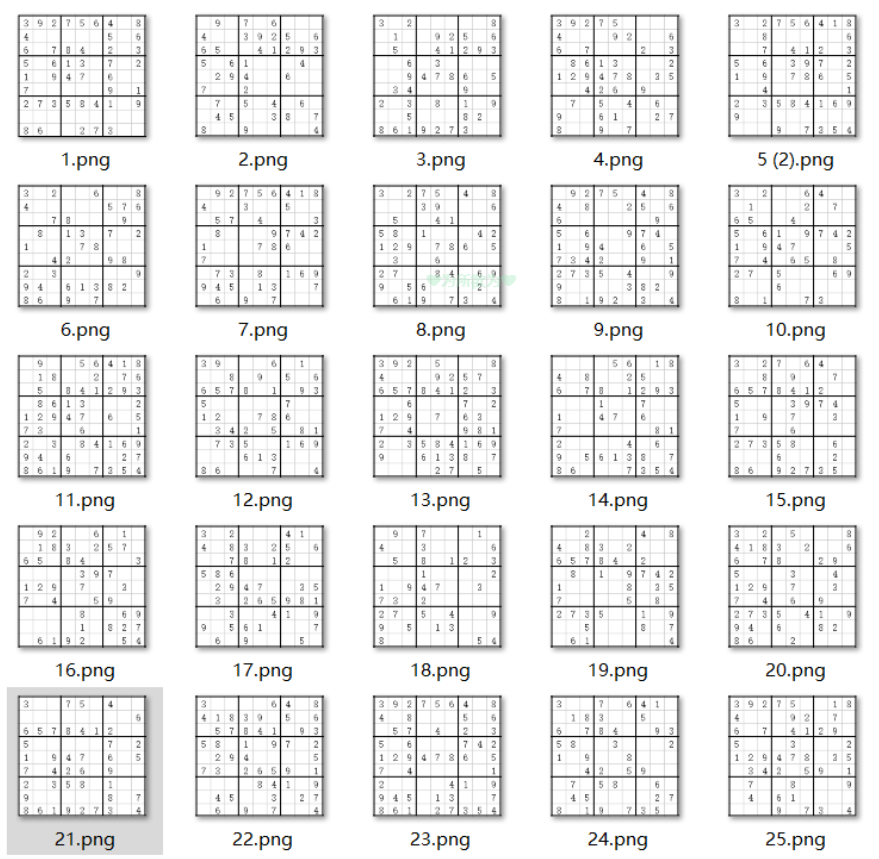
1. 建议使用PS工具，比较好用，将原图和定位角的图适当放大，方便定位。

3、首先需要将定位角的大小缩放适中，适中的标准是：保证定位角的四周都有留白，然后拖动一个定位角的位置，保证：距离两边的像素点有一定空白，且和两边的距离基本一致，其余三个只需按照第一个的位置一一对齐即可。

如果仍然扫不出来，重复2，重复2几次没用后重复1。

工具附上：ps、二维码在线生成网站cli.im

题目附上：ctf.bugku.com/challenges/detail/id/33.html

下载下是一个没有后缀的文件，使用winhex打开，头文件为50 4b 03 为zip文件，修改后缀，打开压缩包，是一大堆数独图片，按照5\*5的格式放置。

想想都知道不可能让你去做数独题，所以转换思路，其中有几张图很想二维码，考虑能否转换为二维码，看到1、5、21三张图做定位符的顺序不对，因此改变顺序，去把5放21，把21放1，把1放5，就行，然后把有数字的涂黑，就能出现二维码了。

也可以进行扒数据，用记事本将图片有数字的记为1，空白的记为0（头疼了，等我记完，天都黑了）

111111101010101000101000001111110000101111111

100000101100111101010011101100011001001000001

101110101110011111010011111101000101001011101

101110101101100010001010000011110001101011101

101110100011100100001111101111111011101011101

100000101100100000011000100001110100001000001

111111101010101010101010101010101011101111111

000000000011001101001000110100110011100000000

110011100100100001111111100100101000000101111

101001001011111111101110101011110101101001100

100000111100100100000110001101001101010001010

001100010011010001010011000100000010110010000

010110101010001111110100011101001110101101111

100011000100011100111011101101100101101110001

001100110100000000010010000111100101101011010

101000001011010111110011011111101001110100011

110111110111011001101100010100001110000100000

110101000010101000011101101101110101101001100

010011111110001011111010001000011011101101100

011001011001010101100011110101001100001010010

010111111111101011111111101101101111111111100

011110001100000100001000101000100100100011110

111110101110011100111010110100110100101010010

110010001011101011101000111100000011100010000

101011111011100111101111111100001010111110010

110100011000111000100111101101111101000100010

111101111110001001000011010110001111110111110

011001010101000110010100010001000101101010001

011101110101101101100100001101101000111101001

110110001001101100010101101111110100101100110

000011100111000000000100001010101111100010010

111010010011110011101110010100001011111010010

101001100010111111110100000100001010101010100

000010011001001101110101001111100101111101101

000010111101110001101011000001000101110100110

011110011010100010100000011011000001110010000

100110100100001101111111101100101110111110011

000000001111110101101000101011100100100011010

111111100011111011011010101101110011101011110

100000101110101101101000111110010001100010001

101110101011100001111111101101001000111111011

101110100110111101101000001001101100011101101

101110100000011101100001101010110010010010001

100000101011001011111011001011000011010110000

111111101010101001111011110101101110000101101

自己写一篇脚本（我是上网找的、懒得写了 ps：压根不会）



生成二维码之后，扫码即可，得到一大串一看就是base64的编码

Vm0xd1NtUXlWa1pPVldoVFlUSlNjRlJVVGtOamJGWnlWMjFHVlUxV1ZqTldNakZIWVcxS1IxTnNhRmhoTVZweVdWUkdXbVZHWkhOWGJGcHBWa1paZWxaclpEUmhNVXBYVW14V2FHVnFRVGs9

这里是base64的多层加密，一层一层的解出来即可，最后得到的flag{y0ud1any1s1}

二维码还有另一种格式，称为汉信码，这种二维码是由中国自主研发出来的适用于汉字的一种二维码，我国拥有完全的自主知识产权。

汉信码具有以下几个显著的特点

1、超强的汉字表示能力（支持GB 18030中规定的160万个汉字信息字符）

2、汉字编码效率高（采用12比特的压缩比率，每个符号可表示12~2174个汉字字符）

3、信息密度高（可以用来表示数字、英文字母、汉字、图像、声音、多媒体等一切可以二进制化的信息）

4、支持加密技术（是第一种在码制中预留加密接口的条码，它可以与各种加密算法和密码协议进行集成，因此具有极强的保密防伪性能）

5、抗污损和畸变能力强（可以被附着在常用的平面或桶装物品上，并且可以在缺失两个定位标的情况下进行识读）

例：

工具附上：http://www.efittech.com/hxdec.html 在线汉信码识别器

http://www.efittech.com/hanxin/cp\_hanxin\_test.aspx汉信码在线生成器

题目附上：2020网鼎杯青龙组 MISC 虚幻2

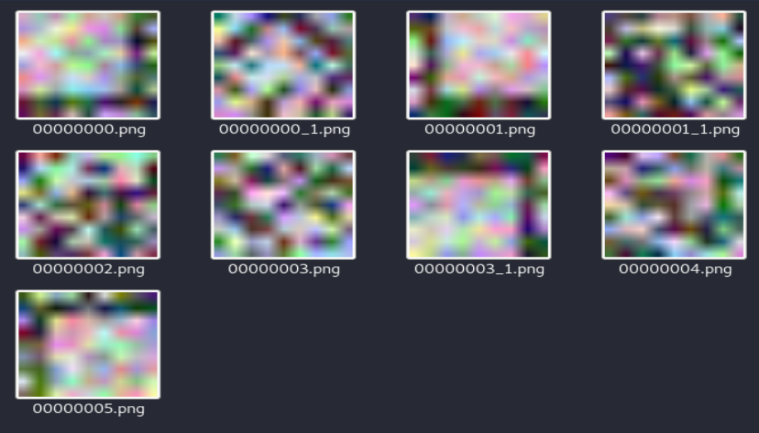
Writeup视频链接（补全汉信码的过程文字不太好表示，感兴趣可以查看以下视频）：

www.bilibili.com/video/BV1AK4y1t7T1/?spm\_id\_from=333.788.videocard.0

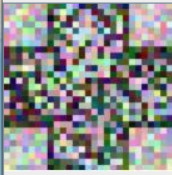
# 补一道题：2018年 网鼎杯CTF

Writeup：

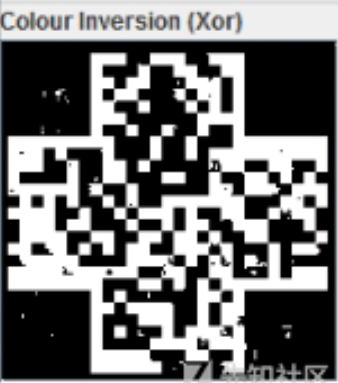
下载附件，是一张小图片，题目提示汉信码。使用 binwalk 提取出 9 张可以用foremost工具分离出来



将图使用ps或其他画图工具拼接成如下，放到steg中翻一下



看着就不像二维码，试着反色一下



网上搜一个汉信码，取定位符，补全，在线扫描得flag

flag{4ab1507d-d195-4d30-87c0-a0d85a77d953}



第九个我们介绍一下emoji编码，emoji 是 Unicode 编码的一部分， 特定形象的Emoji表情符号对应到特定的Unicode字节。  
 常见的Emoji表情符号在Unicode字符集中的范围和具体的字节映射关系, 可通过[Emoji Unicode Tables](https://link.segmentfault.com/?url=http://apps.timwhitlock.info/emoji/tables/unicode%23block-6c-other-additional-symbols" \t "https://segmentfault.com/a/_blank)查看到。

emoji（绘文字）是一种图形字符，网络中经常用作表情符号。emoji不是图片，每个emoji都像文字一样拥有独立编码并且可以存放于字库中，所以可以理解其为图形文字，实际使用中也是和文字一样的使用，可以复制粘贴和输入。

emoji流行于网络中，较新的电脑和手机系统基本都支持显示emoji符号，它们大多拥有独立的emoji图形字库文件，可能包含的数量有所不同。

编码方面，国标码GB18030和国际码Unicode中都有收录emoji图形符号。在Unicode编码中，emoji主要安排在[1号平面第241行至第247行（1F000-1F6FF）](https://www.qqxiuzi.cn/zh/unicode-zifu.php?plane=1&ks=1F000&js=1FFFF)，以及[0号平面第39行和40行（2600-27FF）](https://www.qqxiuzi.cn/zh/unicode-zifu.php?plane=0&ks=2000&js=2FFF)等位置。

emoji图形文字和普通文字一样可复制粘贴，如果输入法有emoji输入功能也可直接输入。可以通过设置字号改变emoji图形大小，但可能无法设置其颜色，因部分emoji的颜色被固化在系统中，例如道路禁止车辆驶入emoji⛔。

工具附上：[Emoji表情符号编码/解码 - 一个工具箱 - 好用的在线工具都在这里！ (atoolbox.net)](http://www.atoolbox.net/Tool.php?Id=937)

www.emojiall.com/zh-hans/blog/340?lang=zh-hans

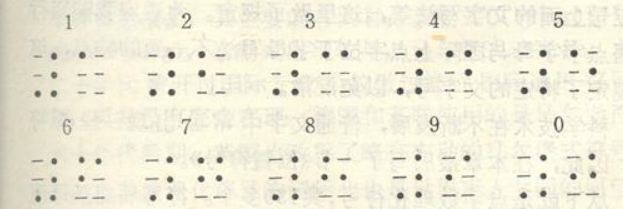
此外杂项中还有诸多奇怪的编码如猪圈密码，栅栏密码，凯撒密码，与佛伦禅，社会主义核心价值观编码，银河文字，盲文，音符，口子码，拼音码，问句码，playfair密码，等等，大多数都能在网上找到对照表或转换器，时间问题不做过多介绍，仅举几个简单的例子。



我们这里给出一些例子做举证：

IMG_256猪圈密**码，对明文X MARKS THE SPOT加密**

盲文数字对照（还有英文和拼音的对照没有搬上来）

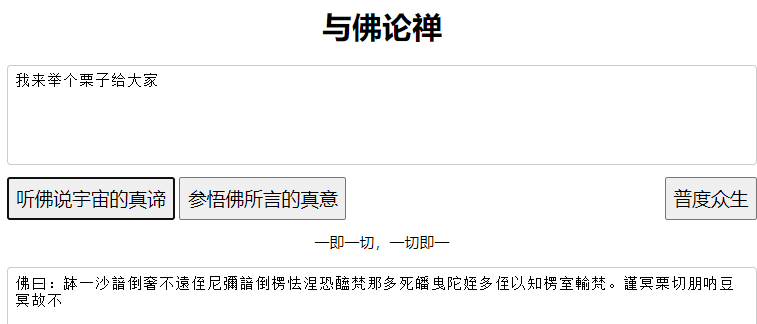


#### 栅栏密码：简单来说，栅栏密码就是把一个明文（去掉空格）分成n组，每组m个，然后一定的排序方法（看下面例子）来将这些字符重新组合。通过m的大小称其为m栏栅栏密码，比较常见的m取2，即2栏栅栏密码。

解码网站：www.qqxiuzi.cn/bianma/zhalanmima.php

例：将have a good night 去掉空格，两个字母为一组分开ha ve ag oo dn ig ht 按竖排组合起来hvaodihaegongt，栅栏密码会有多种排列方式，但万变不离其中，无非就是取的字母组合和排列顺序不通，一般题目中会给到提示。

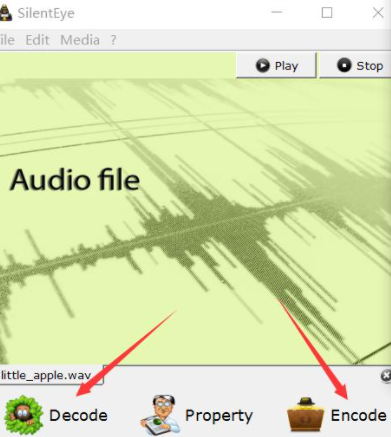
与佛伦禅：www.keyfc.net/bbs/tools/tudoucode.aspx



社会主义核心价值观：[社会主义核心价值观加密解密 (duoluosb.com)](http://z.duoluosb.com/)

1. **音频隐写**

本块内容的知识点会结合题目一起给出，音频隐写中常用的一些工具有[Silenteye](http://silenteye.v1kings.io/" \t "https://www.freesion.com/article/9570534912/_blank)、Mp3stego、Audacity等工具，其中[Silenteye](http://silenteye.v1kings.io/" \t "https://www.freesion.com/article/9570534912/_blank)是一款可以跨平台的应用程序设计，可以使用隐写术，将消息隐藏到图片或声音中。它提供了一个很好的界面，并通过使用插件系统轻松集成了新的隐写算法和加密过程。在广东省强网杯-2015: Little Apple中有它的使用，非常简单，就是考你知不知道这个软件，不知道就没办法做，用软件打开即可。

* 

1、MP3隐写

在压缩过程中，MP3Stego 会将信息隐藏在 MP3 文件中。数据首先被压缩、加密，然后隐藏在 MP3 比特流中。

题目附上：ISCC-2016：Music Never Sleep

工具附上：[MP3stego](https://www.petitcolas.net/steganography/mp3stego/" \t "https://www.sqlsec.com/2018/01/_blank)

Writerup：

首先听音频未发现异常后使用mp3stego打开mp3文件，搜索关键字ctf

IMG_256

得到密码，使用mp3stego进行解密

得到文件 iscc2016.mp3.txt

这里一看就是base系列，先后使用base64和base32解密，得出flag

本题主要考察软件mp3stego的使用，以及对base系列数据的处理

2、频谱图

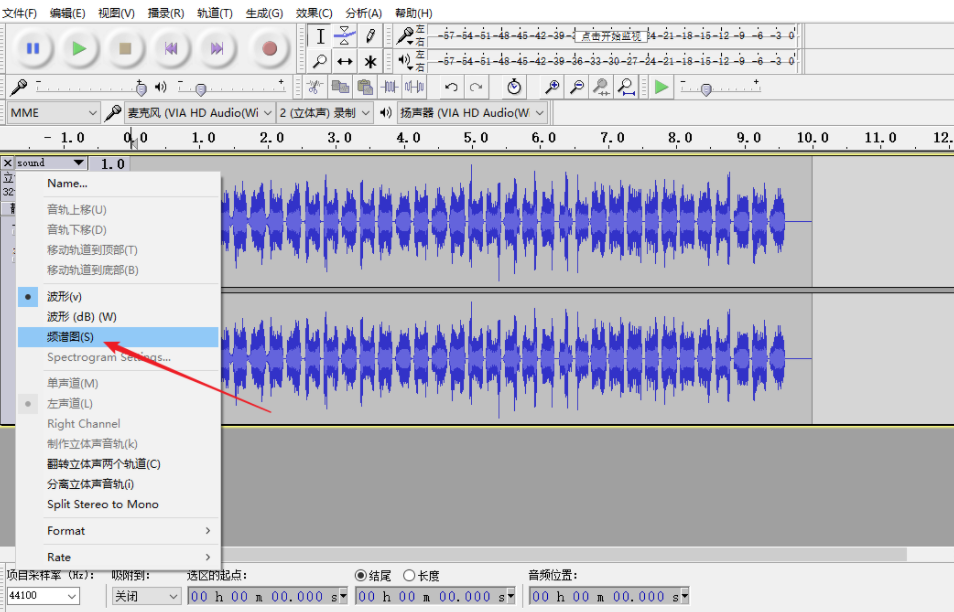
主要将一些信息隐藏在音波频率中，一班=般可以查看频谱图发现一部分信息，能直接看到flag的很少见，主要使用Audacity工具即可，难度不大。主要考察WAV 音频文件隐写术和Audacity的基础使用。

题目附上：Sharif University Quals CTF 2014: Hear with your Eyes

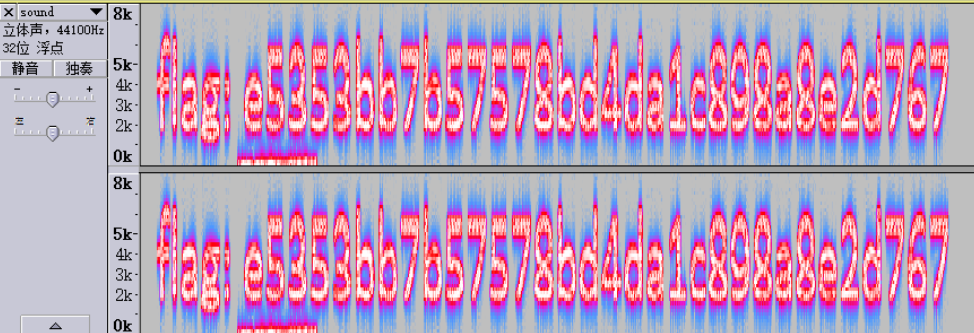
工具附上：Audacity

Writerup：

既然题目给了用眼睛去倾听，那就用Audacity打开wav文件，直接先查看频谱图。



Flag直接得出



3、摩斯编码

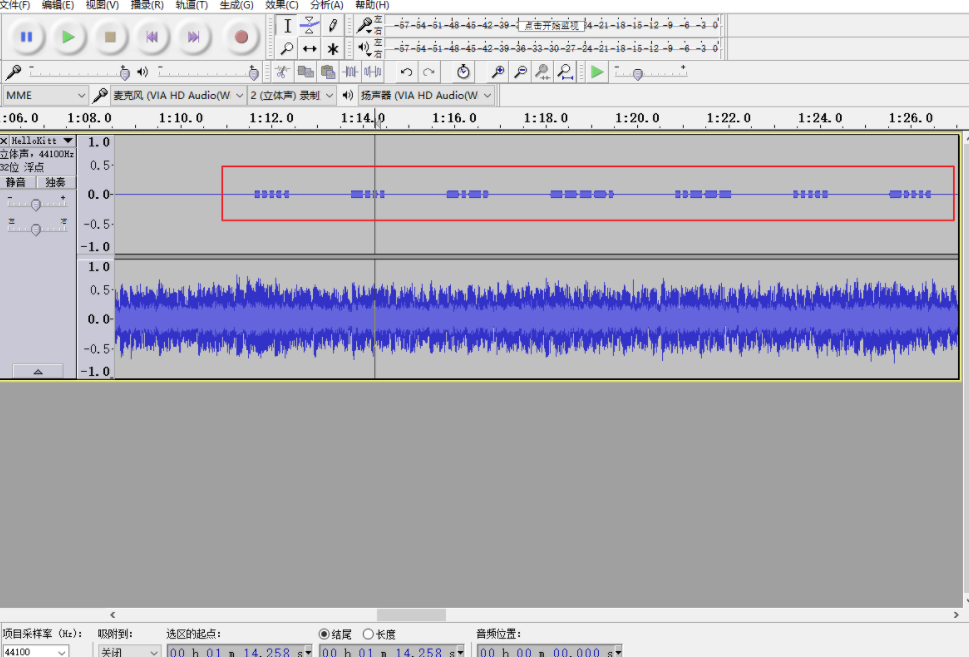
就是把摩斯密码改成了声音，音波，或者频率图，只需一部转换即可得到标准的摩斯密码，使用python语言会较为方便。

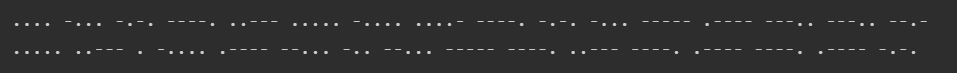
题目附上：Nuit du Hack CTF Qualifications: Here, kitty kitty!

工具附上：Audacity

Writeup：

首先使用工具打开音频，在图中可以看到右上角的音频特殊，且规律很像摩斯密码。



可以看出短线是.长线是-，翻译如下

可以在线解码，也可以使用python得flag

进行md5的小写解密得出flag

4、波形图

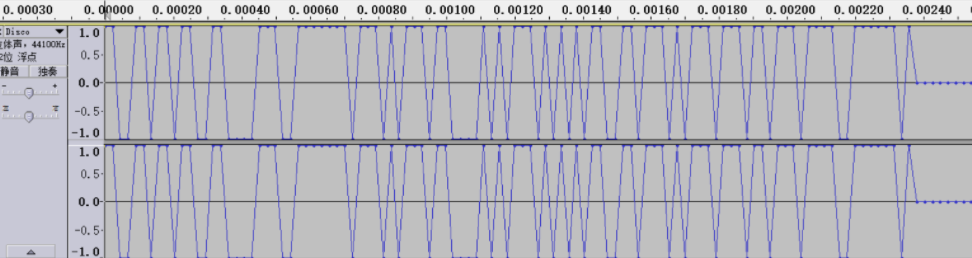
与频谱图相似，但是要略复杂一些

题目附上：ISCC 2017：普通的DISCO

工具附上：Audacity

Writeup：

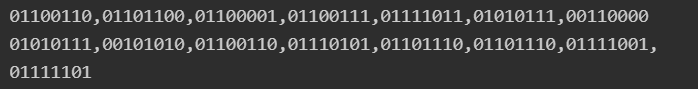
使用工具打开wav文件后，放大波形图



以高为1低为0，转换得到01字符串

110011011011001100001110011111110111010111011000010101110101010110011011101011101110110111011110011111101

一共105位，不符合8位一个字符，符合7位，于是在每个7位之前加个0，得



然后二进制转十进制再转ASCII码即可得到flag

flag{W0W\*funny}

本题考察的波形图并不难，难点在于要想到105位的字符分开不符合一个字符八位，按七位分开后在前面加0，需要有一定的数据处理能力。