

*Forensics & Steganography

COMPASS CTF

October 9, 2021

Last week we have tutorial for web challenges. Various materials are required and learned. Please take some time to review the contents:

Outline:

- * Toolkit
 - * 虚拟化分析环境
 - * 静态分析工具
 - * 动态分析工具
- * Web Tutorial
 - * HTTP/HTTPS协议
 - * OWASP TOP 10
- * Database Sketch
 - * MySQL语法与注入
 - * NoSQL注入
- * Challenges

Forensics is the art of recovering the digital trail left on a computer. There are plenty of methods to find data which is seemingly deleted, not stored, or worse, covertly recorded.

- ✦ 隐写
 - ✦ 图片隐写
 - ✦ 音频/视频隐写
 - ✦ OFFICE文档
 - ✦ 流量包/交换数据流
 - ✦ 其他: HTML/Vmdk/字节码

- * 取证分析
 - * 流量包取证分析
 - * 内存/磁盘镜像分析
 - * 压缩包/固件
- * 编码
 - * 进制编码（十六进制）
 - * Base编码/URL编码
 - * 图片码
 - * 莫尔斯电码/ShellCode/敲击码
 - * 密码学编码

电话拨号编码 (DTMF)

1-9 分别使用 1-9 个脉冲, 0 则表示使用 10 个脉冲。

<https://onlinetonegenerator.com/dtmf.html>

Morse 编码

国际摩尔斯电码

1. 一点的长度是一个单位。
2. 一划是三个单位。
3. 在一个字母中点划之间的间隔是一点。
4. 两个字母之间的间隔是三点 (一划)。
5. 两个单词之间的间隔是七点。

| | | | |
|---|---------|---|-----------|
| A | • — | U | •• — |
| B | ••• — | V | •• — — |
| C | • — •• | W | • — — — |
| D | • — •• | X | •• — — |
| E | • | Y | • — — • |
| F | •• — — | Z | — — •• |
| G | • — — | | |
| H | •••• | | |
| I | •• | | |
| J | • — — — | | |
| K | • — — — | 1 | • — — — — |
| L | • — •• | 2 | •• — — — |
| M | — — | 3 | ••• — — |
| N | • — | 4 | •••• — |
| O | — — — | 5 | ••••• |
| P | • — — • | 6 | ••••• — |
| Q | • — — • | 7 | •••••• |
| R | • — •• | 8 | ••••••• |
| S | ••• | 9 | ••••••• — |
| T | — | 0 | — — — — — |

由01串构成, 或写成.-形式, 除去常规字母数字之外, 也可以用来编码Unicode字符

<https://gchq.github.io/CyberChef/>

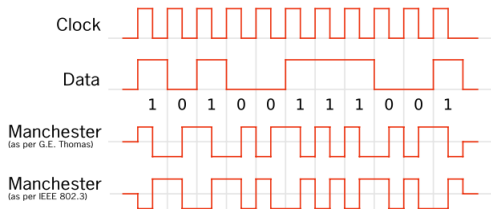
敲击码

敲击码（**Tap code**）是一种以非常简单的方式对文本信息进行编码的方法。因该编码对信息通过使用一系列的点击声音来编码而命名，敲击码是基于 5×5 方格波利比奥斯方阵来实现的，不同点是是用 **K** 字母被整合到 **C** 中。

| Tap Code | 1 | 2 | 3 | 4 | 5 |
|----------|---|---|-----|---|---|
| 1 | A | B | C/K | D | E |
| 2 | F | G | H | I | J |
| 3 | L | M | N | O | P |
| 4 | Q | R | S | T | U |
| 5 | V | W | X | Y | Z |

曼彻斯特编码

能够用信号的变化来保持发送设备和接收设备之间的同步。它用电压的变化来分辨0和1，从高电平到低电平的跳变代表1，而从低电平到高电平的跳变代表0(as per G.E.Tomas编码方式)。从高电平到低电平的跳变代表0，而从低电平到高电平的跳变代表1(as per IEEE 802.3编码方式)。



字母表编码

又称A1Z26编码，部分邪道题目会搞出A26Z1一类的操作。

ASCII 编码

不会吧不会吧不会有人还不知道ASCII编码吧

ASCII编码和上文提到的字母表编码都可以用Cyberchef转换，当然用多了是会记住不少ASCII编码对应的。

- * 0-9, 48-57
- * A-Z, 65-90
- * a-z, 97-122

进制编码可以看作是ASCII编码对应的数字转化为特定数字的形式。

其中16进制解码采用每两个Hex字符对应一个数字，依次进行转化。例如：
434f4d50415353 -> 43 4f 4d 50 41 53 53 -> COMPASS

Base 编码

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。

The Base64 Alphabet

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

- ✦ 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

要注意JWT令牌也采用BASE64编码，可以依次还原出令牌的前两段（第三段为加密签名的BASE64形式）

许多密钥或密码学字符串也会使用BASE64编码后保存（因为BASE64作为一种可打印的编码）

霍夫曼编码

霍夫曼编码使用变长编码表对源符号（如文件中的一个字母）进行编码，其中变长编码表是通过一种评估来源符号出现概率的方法得到的，出现概率高的字母使用较短的编码，反之出现概率低的则使用较长的编码，这便使编码之后的字符串的平均长度、期望值降低，从而达到无损压缩数据的目的。

霍夫曼树又称最优二叉树，是一种带权路径长度最短的二叉树。所谓树的带权路径长度，就是树中所有的叶结点的权值乘上其到根结点的路径长度（若根结点为0层，叶结点到根结点的路径长度为叶结点的层数）。树的路径长度是从树根到每一结点的路径长度之和，记为 $WPL = (W_1 * L_1 + W_2 * L_2 + W_3 * L_3 + \dots + W_n * L_n)$ ， N 个权值 W_i ($i=1,2,\dots,n$) 构成一棵有 N 个叶结点的二叉树，相应的叶结点的路径长度为 L_i ($i=1,2,\dots,n$)。可以证明霍夫曼树的 WPL 是最小的。

XXencoding

XXencode 将输入文本以每三个字节为单位进行编码。如果最后剩下的资料少于三个字节，不够的部份用零补齐。这三个字节共有 24 个 Bit，以 6bit 为单位分为 4 个组，每个组以十进制来表示所出现的数值只会落在 0 到 63 之间。以所对应值的位置字符代替。

| 1 | 2 | 3 | 4 | 5 | 6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0123456789012345678901234567890123456789012345678901234567890123 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| +0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 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 | 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 |

URL 编码

| | | | | | | | | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| ! | # | \$ | & | ' | (|) | * | + | , | / | : | ; | = | ? | @ | [|] |
| %21 | %23 | %24 | %26 | %27 | %28 | %29 | %2A | %2B | %2C | %2F | %3A | %3B | %3D | %3F | %40 | %5B | %5D |

Unicode 编码

为解决ASCII字符集有限的限制，因而引入了长度更长的Unicode编码。

- * `&#x [Hex]: The`
- * `&# [Decimal]: The`
- * `\U [Hex]: \U0054\U0068\U0065`
- * `\U+ [Hex]: \U+0054\U+0068\U+0065`

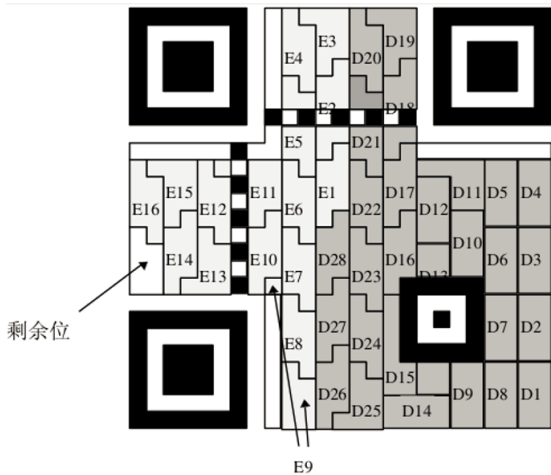
条形码

- * 宽度不等的多个黑条和空白，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

<https://online-barcode-reader.inliteresearch.com/>

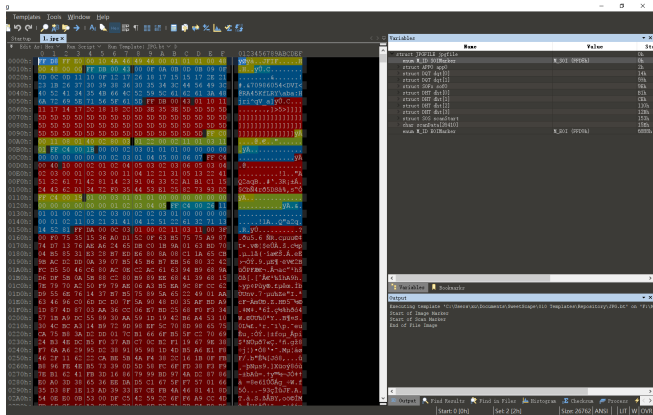
二维码

用某种特定几何图形按一定规律在平面分步的黑白相间的图形记录数据符号信息



010 Editor

SweetScape 010 Editor 是一个全新的十六进制文件编辑器，它有别于传统的十六进制编辑器在于它可用「范本」来解析二进制文件，从而让你读懂和编辑它。它还可用来比较一切可视的二进制文件。



file 命令

file 命令根据文件头（魔法字节）去识别一个文件的文件类型。

```
root in ~/Desktop/tmp λ file flag
flag: PNG image data, 450 x 450, 8-bit grayscale, non-interlaced
```

strings 命令

打印文件中可打印的字符，经常用来发现文件中的一些提示信息或是一些特殊的编码信息，常常用来发现题目的突破口。

Solution

```
strings test | grep -i XXCTF
```

binwalk / foremost

binwalk 本是一个固件的分析工具，比赛中常用来发现多个文件粘合再在一起的情况。根据文件头去识别一个文件中夹杂的其他文件，有时也会存在误报率（尤其是对 Pcap 流量包等文件时）。

```
root in ~/Desktop/tmp λ binwalk flag
```

| DECIMAL | HEXADECIMAL | DESCRIPTION |
|---------|-------------|---|
| 0 | 0x0 | PNG image, 450 x 450, 8-bit grayscale, no |
| 134 | 0x86 | Zlib compressed data, best compression |
| 25683 | 0x6453 | Zip archive data, at least v2.0 to extrac |
| 26398 | 0x671E | Zip archive data, at least v2.0 to extrac |
| 457387 | 0x6FAAB | End of Zip archive |

配合 `-e` 参数可以进行自动化提取。

也可以结合 `dd` 命令进行手动切割。

元数据 (Metadata)

元数据 (Metadata)，又称中介数据、中继数据，为描述数据的数据 (Data about data)，主要是描述数据属性 (property) 的信息，用来支持如指示存储位置、历史数据、资源查找、文件记录等功能。

PNG

文件头 89 50 4E 47 0D 0A 1A 0A + 数据块 + 数据块 + 数据块.....

PNG 定义了两种类型的数据块，一种是称为关键数据块 (critical chunk)，这是标准的数据块，另一种叫做辅助数据块 (ancillary chunks)，这是可选的数据块。关键数据块定义了 4 个标准数据块，每个 PNG 文件都必须包含它们，PNG 读写软件也都必须要支持这些数据块。

例题：调整图像宽度恢复图像

LSB

LSB 全称 **Least Significant Bit**，最低有效位。PNG 文件中的图像像数一般是由 RGB 三原色（红绿蓝）组成，每一种颜色占用 8 位，取值范围为 0x00 至 0xFF，即有 256 种颜色，一共包含了 256 的 3 次方的颜色，即 16777216 种颜色。

而人类的眼睛可以区分约 1000 万种不同的颜色，意味着人类的眼睛无法区分余下的颜色大约有 6777216 种。

LSB 隐写就是修改 RGB 颜色分量的最低二进制位（LSB），每个颜色会有 8 bit，LSB 隐写就是修改了像数中的最低的 1 bit，而人类的眼睛不会注意到这前后的变化，每个像素可以携带 3 比特的信息。

<http://www.caesum.com/handbook/Stegsolve.jar>

JPG

JPEG 是有损压缩格式，将像素信息用 JPEG 保存成文件再读取出来，其中某些像素值会有少许变化。在保存时有个质量参数可在 0 至 100 之间选择，参数越大图片就越保真，但图片的体积也就越大。一般情况下选择 70 或 80 就足够了

JPEG 没有透明度信息

JPG 基本数据结构为两大类型：「段」和经过压缩编码的图像数据。

<https://github.com/redNixon/stegdetect>

<http://linux01.gwdg.de/~alatham/stego.html>

<http://silenteye.v1kings.io/>

GIF

- * 文件头 (File Header)
 - * GIF 文件署名 (Signature)
 - * 版本号 (Version)
- * GIF 数据流 (GIF Data Stream)
 - * 控制标识符
 - * 图象块 (Image Block)
 - * 其他的一些扩展块
 - * 文件终结器 (Trailer)

GIF 文件每一帧间的时间间隔也可以作为信息隐藏的载体。

与音频相关的 CTF 题目主要使用了隐写的策略，主要分为 MP3 隐写，LSB 隐写，波形隐写，频谱隐写等等。

MP3 隐写

MP3 隐写主要是使用 Mp3Stego 工具进行隐写

<http://www.petitcolas.net/steganography/mp3stego/>

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。

频谱

音频中的频谱隐写是将字符串隐藏在频谱中，此类音频通常会有一个较明显的特征，听起来是一段杂音或者比较刺耳。



LSB 音频隐写

类似于图片隐写中的 LSB 隐写，音频中也有对应的 LSB 隐写。主要可以使用 Silenteye 工具

<http://silenteye.v1kings.io/>

CTF 比赛中，流量包的取证分析是另一项重要的考察方向。

通常比赛中会提供一个包含流量数据的 PCAP 文件，有时候也会需要选手们先进行修复或重构传输文件后，再进行分析。

PCAP 这一块作为重点考察方向，复杂的地方在于数据包里充满着大量无关的流量信息，因此如何分类和过滤数据是参赛者需要完成的工作。

- * 流量包修复
- * 协议分析
- * 数据提取

PCAP 文件结构

一般来说, 对于 PCAP 文件格式考察较少, 且通常都能借助于现成的工具如 `pcapfix` 直接修复

<https://f00l.de/hacking/pcapfix.php>

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                               Block Type                               |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                               Block Total Length                       |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
/                               Block Body                               /
/                               /* variable length, aligned to 32 bits */ /
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                               Block Total Length                       |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

协议分析

网络协议为计算机网络中进行数据交换而建立的规则、标准或约定的集合。例如，网络中一个微机用户和一个大型主机的操作员进行通信，由于这两个数据终端所用字符集不同，因此操作员所输入的命令彼此不认识。为了能进行通信，规定每个终端都要将各自字符集中的字符先变换为标准字符集的字符后，才进入网络传送，到达目的终端之后，再变换为该终端字符集的字符。当然，对于不相容终端，除了需变换字符集字符外还需转换其他特性，如显示格式、行长、行数、屏幕滚动方式等也需作相应的变换。

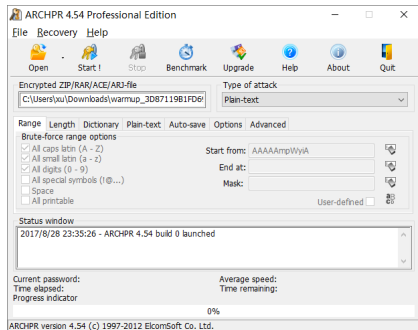
Wireshark: <http://blog.jobbole.com/73482/>

数据提取

ts shark 作为 wireshark 的命令行版，高效快捷是它的优点，配合其余命令行工具 (awk,grep) 等灵活使用，可以快速定位，提取数据从而省去了繁杂的脚本编写

ZIP 文件主要由三部分构成，分别为
压缩源文件数据区 核心目录 目录结束
爆破

<http://www.downcc.com/soft/130539.html>



<https://github.com/hyc/fcrackzip>

CRC32

CRC 本身是「冗余校验码」的意思，CRC32 则表示会产生一个 32 bit (8 位十六进制数) 的校验值。由于 CRC32 产生校验值时源数据块的每一个 bit (位) 都参与了计算，所以数据块中即使只有一位发生了变化，也会得到不同的 CRC32 值。

CRC32 校验码出现在很多文件中比如 png 文件，同样 zip 中也有 CRC32 校验码。值得注意的是 zip 中的 CRC32 是未加密文件的校验值。

这也就导致了基于 CRC32 的攻击手法。

文件内内容很少 (一般比赛中大多为 4 字节左右)

加密的密码很长

我们不去爆破压缩包的密码，而是直接去爆破源文件的内容 (一般都是可见的字符串)，从而获取想要的信息。

明文攻击

一个加密的压缩文件

压缩文件的压缩工具，比如 2345 好压，WinRAR，7z。zip 版本号等，可以通过文件属性了解。如果是 Linux 平台，用 `zipinfo -v` 可以查看一个 zip 包的详细信息，包括加密算法等

知道压缩包里某个文件的部分连续内容 (至少 12 字节)

如果你已经知道加密文件的部分内容，比如在某个网站上发现了它的 `readme.txt` 文件，你就可以开始尝试破解了。

首先，将这个明文文件打包成 zip 包，比如将 `readme.txt` 打包成 `readme.zip`。

打包完成后，需要确认二者采用的压缩算法相同。一个简单的判断方法是用 WinRAR 打开文件，同一个文件压缩后的体积是否相同。如果相同，基本可以说明你用的压缩算法是正确的。如果不同，就尝试另一种压缩算法。

<http://www.downcc.com/soft/130539.html>

伪加密

在上文 ZIP 格式中的 核心目录区 中，我们强调了一个叫做通用位标记 (General purpose bit flag) 的 2 字节，不同比特位有着不同的含义。

| | | | | | |
|-------------------------------|-----------------|---------------|-----------|--------------|----------|
| struct ZIPFILELOCAL_ENTRY | 1 pos | 0h | 00000 | 1 bit | 0 |
| struct ZIPFILEENTRY_DIRECTORY | 1 pos | 00000h | 57h | 1 bit | 0 |
| char deSignature[] | PK | 00000h | 4h | 1 bit | 0 |
| ushort deVersionMadeBy | 31 | 0000Ah | 2h | 1 bit | 0 |
| ushort deVersionOfExtract | 20 | 0000Ch | 2h | 1 bit | 0 |
| ushort deFlags | 0 | 00000h | 2h | 1 bit | 0 |
| enum COMPRESSION_METHOD | COMP_DEFLATE_00 | 00000h | 2h | 1 bit | 0 |
| DOSDATE deDate | 224136 | 00002h | 2h | 1 bit | 0 |
| DOSDATE deFileDate | 09282017 | 00004h | 2h | 1 bit | 0 |
| uint deCrc | 8A12170h | 00006h | 4h | 1 bit | 0 |
| uint deCompressedSize | 22469 | 0000Ah | 4h | 1 bit | 0 |
| uint deUncompressedSize | 47419 | 0000Ch | 4h | 1 bit | 0 |
| ushort deFileNameLength | 5 | 00002h | 2h | 1 bit | 0 |
| ushort deExtraFieldLength | 36 | 00006h | 2h | 1 bit | 0 |
| ushort deFileCommentLength | 0 | 00000h | 2h | 1 bit | 0 |
| ushort deDiskNumberStart | 0 | 00000h | 2h | 1 bit | 0 |
| ushort deInternalAttributes | 0 | 0000Ah | 2h | 1 bit | 0 |

常见的磁盘分区格式有以下几种

Windows: FAT12 -> FAT16 -> FAT32 -> NTFS

Linux: EXT2 -> EXT3 -> EXT4

FAT 主磁盘结构

VMDK

VMDK 文件本质上是物理硬盘的虚拟版，也会存在跟物理硬盘的分区和扇区中类似的填充区域，我们可以利用这些填充区域来把我们需要隐藏的数据隐藏到里面去，这样可以避免隐藏的文件增加了 VMDK 文件的大小（如直接附加到文件后端），也可以避免由于 VMDK 文件大小的改变所带来的可能导致的虚拟机错误。而且 VMDK 文件一般比较大，适合用于隐藏大文件。

<https://medium.com/@zemelusa/first-steps-to-volatile-memory-analysis-dcbd4d2d56a1>

pyc 文件

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。

最开始 4 个字节是一个 `Maigc int`，标识此 `pyc` 的版本信息

接下来四个字节还是个 `int`，是 `pyc` 产生的时间

序列化的 `PyCodeObject`，结构参照 `include/code.h`，序列化方法 `python/marshal`

<https://github.com/zrax/pycdc>

<https://github.com/AngelKitty/stegosaurus>

Wireshark doo dooo do doo...

Challenge

8 Solves



Wireshark doo dooo do
doo...

5

Can you find the flag? [shark1.pcapng](#)

Flag

Submit

Glory of the Garden

Challenge

10 Solves



Glory of the Garden

5

This [garden](#) contains more than it seems.

View Hint

Flag

Submit

MacroHard WeakEdge

Challenge

7 Solves

x

MacroHard WeakEdge

5


I've hidden a flag in this file. Can you find it?

[Forensics_is_fun.pptm](#)

Milkslap

Challenge 8 Solves ×

Milkslap
10



View Hint

Flag Submit

Disk, disk, sleuth!

Challenge

6 Solves

X

Disk, disk, sleuth!

10

Use `srch_strings` from the sleuthkit and some terminal-fu to find a flag in this disk image: dds1-alpine.flag.img.gz

View Hint

View Hint

View Hint

View Hint

Flag

Submit

advanced-potion-making

Challenge

8 Solves



advanced-potion-making

15

Ron just found his own copy of advanced potion making, but its been corrupted by some kind of spell. Help him recover it!

 advanced-p...

Flag

Submit

Disk, disk, sleuth! II

Challenge

7 Solves



Disk, disk, sleuth! II

15

All we know is the file with the flag is named `down-at-the-bottom.txt`... Disk image: [dds2-alpine.flag.img.gz](#)

View Hint

View Hint

View Hint

Flag

Submit

So Meta

Challenge

7 Solves

x

So Meta 15

Find the flag in this [picture](#).

View Hint

View Hint

Flag

Submit