CTF 取证类题目指南

取证类题目

在 CTF 中，取证赛题包括了文件分析、隐写、内存镜像分析和流量抓包分析。任何要求检查一个静态数据文件（与可执行程序和远程服务器不同）从而获取隐藏信息的都可以被认为是取证题（除非它包含了密码学知识而被认为是密码类赛题）。

取证作为 CTF 中的一大类题目，不完全包括安全产业中的实际工作，常见的与之相关的工作是事故相应。但即使在事故响应工作中，计算机取证也经常是执法部门获取证据数据和证物的工作，而非对防御攻击者或恢复系统感兴趣的商业事故相应企业。

与大多数 CTF 取证题目不同，现实生活中的计算机取证任务很少会涉及巧妙的编码、隐藏数据、层层嵌套的文件中的文件，或是其他脑洞类的谜题。很多时候刑事案件需要的是精心恢复一个被破坏的 PNG 文件，根据一张照片或 QR 码来解码获取包含 NES 只读内存镜像来输出证据的压缩包密码。也就是说，现实的取证需要从业者能够找出间接的恶意行为证据：攻击者攻击系统的痕迹，或是内部威胁行为的痕迹。实际工作中计算机取证大部分是从日志、内存、文件系统中找出犯罪线索，并找出与文件或文件系统中数据的关系。而网络（流量抓包）取证比起内容数据的分析，更注重元数据的分析，也就是当前不同端点间常用 TLS 加密的网络会话。

CTF 竞赛中的取证类谜题和实际生活中的工作关联较少的原因可能在于这类题目并不像漏洞利用一类的题目受到人们的注意。也有可能是它很少吸引黑客们来参与解答此类题目。不管怎么样，许多参赛者都喜欢解答 CTF 中各种简单多样化的取证题目，考虑到大多数参赛者没有能够在可执行文件分析大赛中带来巨大优势的售价 5000 美元的带有 Hex-Rays 反编译器的 IDA Pro 专业版，这类题目还是相对适合入门新手来做的。

需要的技术

下列三种能力是解答取证类 CTF 题目中最常用到的：

- 掌握一门脚本语言（例如 python）
- 掌握如何使用该语言来操作二进制数据（涉及到字节的操作）
- 认识各类文件格式、协议、结构和编码

前两类技术你可以在 CTF 外自行学习，而第三种则只能在实战中不断熟练。好在通过本文档，你可以有个大致的了解。

当然了，在大部分 CTF 比赛中最常用的环境是 Linux 系统，有时候也会用到 VM 虚拟机中的 Windows。MacOS 也可以拿来代替 Linux，前提是你得忍受部分开源工具无法正确编译运行。

通过 python 操作二进制数据

假设你已经初步掌握了一些 Python 编程技巧，你也可能不太清楚如何有效地操作二进制数据。类似 C 的低级语言可能更适合这一工作，但 Python 开源社区中的各种包比起学习 C 语言的难度，可以帮助你更快速的上手操作二进制数据。

下面是常用的 python 操作二进制数据的例子

在二进制模式中读取和写入文件：

```
f = open('Reverseit', "rb")
s = f.read()
f.close()
f = open('ItsReversed', "wb")
f.write(s[::-1])
f.close()
```

字节数组是一种数组的可变序列，在 Python2 和 3 都可以使用。

```
>>> s = bytearray(b"Hello World")
>>> for c in s: print(c)
...
72
101
108
108
111
32
87
111
114
108
100
```

你也可以通过 16 进制的字节数组来表示 Unicode 字符串：

```
>>> example2 = bytearray.fromhex(u'00 ff')
>>> example2
bytearray(b'\x00\xff')
>>> example2[1]
```

字节数组的格式包括了大多数 python 中常用的字符串和列表函数，例如：split(), insert(), reverse(), extend(), pop(), remove().

将文件以二进制数组形式读取的例子：

```
data = bytearray(open('challenge.png', 'rb').read())
```

常见的取证概念和工具

接下来会讲述常见的 CTF 取证题目概念和一些帮助初步解题的推荐工具。

文件格式的判断（以及’魔法字节’）

几乎所有的取证题目都会涉及到一个有时候连告诉你是什么文件类型的提示都没有的文件。文件类型对于用户来说长期是和文件扩展名相关联的（例如 MarkDown 的 readme.md 文件）MIME 类型（互联网上指定应用程序打开对应扩展名的协议）或是存储在文件系统上的元数据（泪如 Mac OS 上的 mdls 命令）。在 CTF 中，有时候是用各种方法来尝试判断文件的类型。

在 UNIX 系统中，传统的识别文件类型方法是 libmagic，也就是识别被称作‘魔法数字’或者‘魔法字节’——文件头中特定字节的库。libmagic 库是文件操作中最基础的命令：

```
$ file screenshot.png
screenshot.png: PNG image data, 1920 x 1080, 8-bit/color RGBA, non-interlaced
```

考虑到这是 CTF 竞赛，因此竞赛时命题人会故意设计一些针对常用工具和方法，故意误导人的文件。同时如果一个文件内部包含了其他文件的信息，那么文件命令只能够识别其中包含的文件类型。此时你需要进一步深入地分析文件内容。

TrID 是一个专业的文件分析工具，尽管它没有开源，但它在多个平台上都是免费使用的。它还允许一定程度上的自主分析功能。它的优势在于内部包含有世上大多数文件类型的属性和混淆格式。

文件雕复

文件中的文件，或是加载在固件和文件系统中的文件是 CTF 中常见的一类取证题目。针对识别文件系统中附加的其他文件并进行提取的技术称作‘文件雕复’。最常用于这一工作的固件分析工具是 binwalk。

scalpel 作为 SluethKit（在文件系统中会具体谈到）是另一个用于文件恢复的文件雕复工具。

要手动提取文件（已知偏移量）的部分信息，你可以使用 dd 命令。许多 16 进制编辑器也提供了复制黏贴部分字节到新建文件中的功能，所以你不需要专门去研究偏移量。

使用 dd 命令从一个偏移量 1335205 的文件中提取 40668937 字节数据的文件雕复例子：

```
$ dd if=./file_with_a_file_in_it.xxx of=./extracted_file.xxx bs=1
skip=1335205 count=40668937
```

尽管上述工具能够满足基本需求，但有时候你还是需要人工利用 Python 的 re 或 regex 模块来编程提取文件中的一小部分信息，从而判断魔法字节，以及 Python 的 zlib 模块来提取 zlib 数据流。

初步分析

刚开始你可能一头雾水，面对着题目文件不知道如何进行下一步。常见的一些方式包括搜索文件中所有的明文字符串，例如 grep 指令搜索特定字符串，bgrep 来搜索非文本形式的数据规律，还有 hexdump 命令。

通过文件偏移来获取所有 ASCII 字符串的例子如下：

```
$ strings -o screenshot.png
     12 IHDR
     36 $iCCPICC Profile
     88 U2EI4HB
...
     767787 IEND
```

Unicode 字符串，例如 UTF-8 会在搜索 ASCII 字符串时出现。但如果要搜索其他编码，可以翻阅帮助文档中关于-e 指令的说明。需要注意许多编码都有可能在取证中出现，但最简单的编码始终占据着主导地位。

搜索 PNG 文件中 PNG 魔法字节的例子

```
$ bgrep 89504e47 screenshot.png
screenshot.png: 00000000
```

使用 hexdump 的例子：

```
$ hexdump -C screenshot.png | less
```

```
00000000  89 50 4e 47 0d 0a 1a 0a  00 00 00 0d 49 48 44 52
|.PNG........IHDR|
00000010  00 00 05 ca 00 00 02 88  08 06 00 00 00 40 3d c9
|.............@=.|
00000020  a4 00 00 18 24 69 43 43  50 49 43 43 20 50 72 6f
|....$iCCPICC Pro|
00000030  66 69 6c 65 00 00 58 85  95 79 09 38 55 5d f8 ef
|file..X..y.8U]..|
00000040  da 67 9f c9 71 0c c7 41  e6 79 9e 87 0c 99 e7 39
|.g..q..A.y.....9|
:
```

hexdump 的优势不在于它是最好的 16 进制编辑器（它绝对不是）而在于你可以用操作命令直接输出 hexdump，或用 grep 命令来输出，还能用字符串格式化命令来设置输出格式。

使用 hexdump 格式化命令以 64 字节整数的 16 进制来输出文件前 50 个字节例子如下：

```
$ hexdump -n 50 -e '"0x%08x " ' screenshot.png
0x474e5089 0x0a1a0a0d 0x0d000000 0x52444849 0xca050000 0x88020000
0x00000608 0xc93d4000 0x180000a4 0x43436924 0x43434950 0x6f725020
0x00006966
```

其他 16 进制镜像操作例子。

二进制文本编码

二进制指的是 1 和 0，但它们经常以文本的形式传输着。因为如果真的要发送一组 101010101 的序列无疑是十分浪费的，所以数据经常先通过一种方式进行编码。这就是所说的二进制到文本编码，一种十分流行的 CTF 赛题。当遇到在分析文件种字符串的情况时，可以尝试将二进制数据编码为文本字符串。

我们之前提到要精通 CTF 取证题目，需要认识各类编码形式。有些一眼就能看出来，例如 Base64 编码有着字母数字和=结尾的特征（如果出现的话）。现在有许多在线 Base64 编码/解码器，你也可以使用 base64 命令：

```
$ echo aGVsbG8gd29ybGQh | base64 -D
hello world!
```

ASCII 编码的 16 进制由于字符特征（0-9，A-F）而十分好认。ASCII 字符本身占据了一段字节（0x00 到 0x7f）所以如果你在文件中找到了类似 68 65 6c 6c 6f 20 77 6f 72 6c 64 21 的字符串，很明显会都是类似 0x60 的特点——这就

是 ASCII 码。从技术角度来说，这个是文本(“hello world!”)通过 ASCII（二进制）编码后以 16 进制（文本）编码获得的。是不是已经晕了？😉

下列多个站点提供了多种在线的编码/解码。在本地上可以尝试使用 xxd 命令。

使用 xxd 将文本从 ASCII 转换为 16 进制编码的例子：

```
$ echo hello world\! | xxd -p
68656c6c6f20776f726c64210a
```

常见的文件格式

我们已经讨论了取证任务中常用的基础概念和工具，现在我们将具体的分类讨论取证题目，以及各种情况下推荐用于分析的工具。

尽管无法准备每一种可能遇到的数据格式，但 CTF 中还是有不少经常出现的格式。如果你能针对下述情况准备号分析工具，你就能应付大多数的取证赛题：

- 压缩包文件（ZIP，TGZ）
- 图片格式文件（JPG，GIF，BMP，PNG）
- 文件流图片（especially EXT4）
- 流量包文件(PCAP，PCAPNG)
- 内存镜像
- PDF
- 视频(尤其是 MP4)或音频(尤其是 WAV，MP3)
- 微软的 OFFICE 文件（RTF，OLE，OOXML）

在一些难度较高的 CTF 竞赛中，命题人会自豪地出一些要求参赛者分析没有现成的公开工具拿来分析的复杂格式文件。你需要了解如何快速找到没见过的格式相关的文档和工具。许多文件格式都可以通过互联网搜索找到相关的具体开源报告，但如果事先了解过相关文档，无疑会受益匪浅。所以我们列出了下列的参考链接。

当分析文件格式时，能够用于识别不同文件格式（模板）的 16 进制编辑器 010 Editor 是十分有用的，另一个类似的开源工具是 Kaitai。还有类似功能的 Wireshark 网络协议分析器可以用来分析特定的多媒体文件（例如 GIF,JPG 和 PNG）。然而这些工具都是分析完整无损的文件，无法有效的用于 CTF 竞赛中要求你重构缺失一定内容的文件。

你也可以查看下由 Ange Albertini 编辑的文件格式可视化说明。

压缩文件

许多 CTF 题目都会压缩一个在 zip,7z,rar,tar 或 tag 文件中，但只有取证题会将压缩包作为题目的一部分。很多时候题目的目标是从一个损坏的压缩包中提取文件，或是在一个没有用到的区域（常规的取证题目）中找到加载的数据。Zip 既是现实中最常用的，也是 CTF 中最常见的。

下面有多个命令行工具来帮助了解 zip 压缩文件的信息：

- 解压缩操作经常会给出 zip 文件无法解压的相关有用信息
- Zipdetails -v 能够根据参数提供多种信息
- Zipinfo 在无需提取的情况下列出了 zip 文件的内容信息
- zip -F input.zip --out output.zip 和 zip -FF input.zip --out output.zip 尝试修复损坏的 zip 文件
- fcrackzip 尝试爆破 zip 压缩包密码（适用于少于 7 个字母的密码）

Zip 文件格式具体说明

在遇到密码保护的 zip 文件时，需要注意它并没有加密文件名和压缩包中的原始文件大小，而加密保护的 RAR 和 7z 文件则无法查看。

在破解加密 zip 时，如果你有加密压缩包其中一个文件的未加密/未压缩副本，你可以尝试进行（点击查看操作过程和学术原理）的明文攻击操作。但新的密码保护压缩文件（如采用 AES-256 而不是 ZIP 加密）没有这一安全隐患。

图像文件的格式分析

CTF 是十分有趣的，而图像文件能够很好地包含黑客文化，因此 CTF 竞赛中经常会出现各种图像文件。图像文件有多种复杂的格式，可以用于各种涉及到元数据、信息丢失和无损压缩、校验、隐写或可视化数据编码的分析解密。

第一步往往是使用 exiftool 检查图像文件的元数据，CTF 中使用的图像文件中 EXIF 信息往往保留着原始图像例如相机、略缩图、注释、GPS 定位等信息。不论这些元数据是否有用，都值得先去看一看。

Exiftool 导出的部分数据例子：

```
$ exiftool screenshot.png
ExifTool Version Number         : 10.53
File Name                       : screenshot.png
Directory                       : .
File Size                       : 750 kB
File Modification Date/Time     : 2017:06:13 22:34:05-04:00
File Access Date/Time           : 2017:06:17 13:19:58-04:00
File Inode Change Date/Time     : 2017:06:13 22:34:05-04:00
File Permissions                : rw-r--r--
```

```
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 1482
Image Height                    : 648
Bit Depth                       : 8
Color Type                      : RGB with Alpha
Compression                     : Deflate/Inflate
...
Primary Platform                : Apple Computer Inc.
CMM Flags                       : Not Embedded, Independent
Device Manufacturer             : APPL
Device Model                    :
...
Exif Image Width                : 1482
Exif Image Height               : 648
Image Size                      : 1482x648
Megapixels                      : 0.960
```

PNG 图片在 CTF 竞赛中可能由于它低损压缩的格式适合在图片中隐藏不可见的数据而十分流行。PNG 图片也可以在 wireshark 中分割开来，你可以尝试使用 pngcheck 来试图修复被破坏的 PNG 图片，如果你需要深入研究 PNG 文件的数据，那么你可以使用 pngtools。

隐写技术指的是将一些秘密数据储存在现实中较为罕见无关数据（明文）中（通过密码学进行有效加密）的技术，但在 CTF 取证题目中，隐写题目十分流行。隐写的明文可以是任何一种数据，但视频文件由于能够允许一定量不引人注意的数据丢失（同样特征可以实现低损压缩）而更适合作为赛题。隐写的难点不光在于如何识别需要提取的隐写技术，还包括如果使用 steganographic 工具来进行提取。针对赛题的文件，我们可以进行一些方法来尝试验证其是否是一个隐写题目。Stegsolve（JAR 下载链接）经常用于各种图片隐写中判断和提取隐藏数据。你也可以尝试 zsteg。

Gimp 提供了转换各类图像文件可视化数据的功能，CTF 命题人经常使用修改过的 Hue/Saturation/Luminance 值或颜色通道来隐藏一段秘密信息。Gimp 还可以用于确认文件是否是一个图像文件，例如你从内存镜像或其他地方获取了一个图片文件，但你不知道它标注像素格式等信息的头文件，那么你可以将这些数据作为‘Raw image data’交给 Gimp 进行各种处理操作。

ImageMagick 工具集可以嵌入在脚本中帮助你快速进行识别、调整、修改、转换图像文件的各类操作，它的比较工具还可以帮助你快速找出两张看似相同的图片中的数据区别。

如果你在写一个自制的图像解析，可以试着导入 Pillow（Python 图像处理库 PTL）。它允许你从动态 GIF 中提取每一帧图像或是从 JPG 图片中提取不同位置的像素——它基本支持所有的图像文件格式。

如果题目是一个 QR 码（2D 条形码）使用 Python 的 qrtools 模块来检查一下，你能通过不到 5 行的 Python 代码来对 QR 码的图像进行解码。当然你也可以用智能手机来扫一扫对单独的一个 QR 码解码。

文件系统分析

有时候，CTF 取证类赛题会提供一个完整的磁盘镜像，参赛者需要具备一定的策略来在这个数据系统中寻找特定的 flag。在计算机取证中，这类策略指的是快速理清内容的能力。没有策略的话只能耗时耗力地查看所有的信息。

加载光驱文件系统镜像的例子：

```
mkdir /mnt/challenge
mount -t iso9660 challengefile /mnt/challenge
```

一旦你加载了文件系统，那么使用 tree 命令来快速查看目录结构从而判断是否有值得你进一步分析的内容是个好主意。

你也许不需要在可见文件系统中寻找特定的文件，而是要在隐藏卷、未使用空间（硬盘中没有划分到任何一部分的空间）或是类似 http://www.nirsoft.net/utils/alternate_data_streams.html 的非文件系统结构中寻找被删除的文件。对于 EXT3 和 EXT4 文件系统来说，你可以使用 extundelete 尝试恢复被删除的文件。对于其他系统，则可以使用 TestDisk 进行恢复丢失的分区表，修复损坏的文件，恢复 FAT 或 NTFS 盘中的删除文件等多种操作。

The Sleuth Kit 和它的 WEB 用户界面‘Autospy’是一个用于文件系统分析的开源工具，它也许过于针对执法部门的工作需求，但在搜索整个硬盘镜像的特定关键字或未使用空间时仍是个十分好用的工具。

专门针对固定功能的低资源系统中加载了驱动的文件系统是一类特别的题目，它们可以使压缩过的单独文件，或是只读文件。Squashfs 是一类非常受欢迎的驱动文件系统，你可以使用 firmware-mod-kit 或 binwalk 来分析其中加载的镜像文件。

抓包（PCAP）文件分析

网络流量经常通过如 tcpdump 或 Wireshark（都基于 libpcap）的软件以 PCAP（抓包）文件的格式来捕捉并保存。通常 CTF 比赛中会提供一个包含一些流量

数据和需要参赛者恢复/重构传输文件或信息的 PCAP 文件。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。

在初步分析中，可以尝试使用 Wiresharks 的静态和会话视图或 capinfos 命令来从高层视图查看报文。Wireshark 以及命令行版本的 tshark 都支持过滤功能也就是说你精通正则的话，可以快速缩小你需要分析的范围。还有个叫做 PacketTotal 的在线工具能够将你上传的不超过 50MB 的 PCAP 文件转换为可视化时间关系以及 SSL 元数据的界面，它还能够高亮文件传输过程，便于你发现可疑的活动。如果你已经知道你要找的是什么，也可以用 ngrep 命令来搜索整个数据报文。

正如同文件雕复指的是识别和提取加载在其他文件中的文件一样，报文雕复指的是从抓到的数据包中提取出文件。有一些昂贵的商业工具专门完成这类工作，但 Xplico 框架作为开源工具，也能满足这一需求。Wireshark 也有一个能够将数据从抓包中提取的功能‘导出对象’（例如：文件—到处对象—HTTP—保存全部），除此之外，你也可以尝试 tcpxtract,Network Miner,Foremost,或是 Snort。

如果你想要编写自己的脚本来直接处理 PCAP 文件，那么推荐用于操作 pcap 的 Python 包 dpkt，你也可以用 Wirepy 来写 Wireshark 的交互。

如果你要尝试修复损坏的 PCAP 文件，这里有一个叫做 PCAPfix 的在线修复 PCAP 文件服务。

注意 PCAP 和 PCAPNG 的不同：这是两种版本的 PCAP 文件格式。PCAPNG 版本较新，因此有工具不支持该类型文件。你可能需要使用 Wireshark 或其他兼容工具将 PCAPNG 文件转换为 PCAP 从而在其他工具中使用。

内存镜像分析

多年来计算机取证经常被看作是文件系统取证，但当攻击者们越来越熟练时，它们开始避开硬盘进行攻击。同时内存快照由于保存了实时环境（系统设置、远程脚本、密码、密钥等）这些无法在硬盘上找到的线索。所以内存快照/镜像取证是这些年事故响应中越来越多的方法。在 CTF 竞赛中，你可能会遇到提供一个内存镜像文件，并让你从中定位和提取一个隐藏文件或信息的题目。

用于内存镜像分析的主流开源框架是 Volatility，Volatility 是一个支持解析外部工具采集的内存奖项（例如 VM 采集的 VMware 内存镜像）的 Python 脚本。在提供内存镜像文件和相关资料（从镜像中获取的系统）后，Volatility 能够开始识别数据的结构，运行进程，密码等资料。它还支持提取多种信息的人工插件拓展功能。

Ethscan 是一个在内存镜像中寻找类似流量数据，并提取为 Wireshark 中查看的 PCAP 文件的工具。它还有插件可以提取 SQL 数据库，Chrome 访问记录，Firefox 访问记录等数据。

PDF 文件分析

PDF 文件是一个非常复杂的文档文件格式，多年来已经出现了多种可以隐藏数据的地方和技巧，因此它在 CTF 取证题中十分流行。NSA 在 2008 年发表过标题为“Adobe PDF 中的隐藏数据和元数据：公开危险和应对措施”。尽管原 URL 已经无法打开，但你仍可以在这里打开副本。Ange Albertini 也在 Github 上保留着一份关于 PDF 文件格式的隐藏小技巧的 Wiki。

PDF 格式类似 HTML 一样有着部分的明文，但其中还包括了许多二进制‘对象’。Didier Steven 写过关于 PDF 格式的具体材料分析，这些二进制‘对象’可以是压缩数据，也可以是加密数据，还可以类似 Javascript 或 Flash 的脚本语言。你可以通过文本编辑器也可以使用类似 Origami 的 PDF 文件编辑器来显示 PDF 的结构。

qpdf 是一个查看 pdf 文件并整理提取信息时十分有用的工具，另一个是 Ruby 中的 Origami 框架。

在搜索 PDF 文件中的隐藏数据时，常见的隐藏地方包括：

- 不可见的图层
- Adobe 的元数据格式‘XMP’
- Adobe 的 XMP 元数据
- PDF 的‘增量生成’功能允许保留用户不可见的前版本信息
- 白色的文字或背景图
- 图片背后的文字信息
- 图层后面被覆盖的另一个图层
- 不显示的注释层。

还有许多 Python 包能够帮助处理 PDF 文件，例如 PeepDF 允许你制作自己的分解脚本。

视频与音频隐写

像图像文件一样，不仅仅因为其趣味，还由于现实中经常运用音频和视频来隐藏数据，因此音频和视频文件也经常出现在 CTF 取证赛题中。正如同图像文件一样，隐写技术可以将秘密信息加载在内容数据上，你需要知道如何查看文件元数据区域来寻找线索。第一步一般是使用 mediainfo 工具（或 exiftool）来查看内容区域和元数据信息。

Audacity 是一个开源的音频文件和波形图处理工具，CTF 命题人也喜欢用它在音频波形（也就是你看到的频谱图，当然你可以用更加专业的 Sonic Visualiser 来查看））中添加文本信息。Audiacity 还允许你进行降低播放速度、倒置播放等操作来帮助你从怀疑有隐藏信息（比如你听到的杂音、电磁音、混乱的声音等）中寻找隐藏的信息。Sox 是另一个用于转换和操作音频文件的命令行工具。

还有种常见的方法是通过检查 LSB 来寻找隐藏信息。许多音频和视频都采用了分离（固定长度）的数据块从而实现流畅播放，这些数据块的 LSB 是一个非常常见的不影响文件肉眼可见层面的隐藏数据的位置。

其他情况还有将信息通过双音多频或莫尔斯码的编码方式隐藏在音频中，针对这些可以使用 multimon-ng 来解码。

视频文件格式由于它采用分离的音频流和视频流而十分适合存储各类信息。推荐使用 ffmpeg 来分析和操作视频文件。Ffmpeg -i 能够提供文件内容的初步分析，也可以用于分离或播放不同的内容流。在 Python 中也可以 ffmpy 实现 ffmpeg 的功能。

Office 文档分析

微软开发了多种 office 文档的文件格式，其中多数由于支持宏（VBA 脚本）而被用于钓鱼攻击和执行恶意软件。微软文档的取证分析与 PDF 的取证分析区别不大，只是更贴近现实中的事故相应。

大致来说存在两类 Office 文件格式：早期的文件格式（例如 RTF，DOC，XLS，PPT 等扩展名）和 Office Open XML 格式（例如 DOCX、XLSX、PPTX 等扩展名）。这两类格式都是支持链接和附加内容（对象）的组合结构二进制文件格式。OOXML 文件实际上是一种压缩包容器（参考上文提及的压缩文件），也就是解压缩文件是最简单地查看隐藏数据的方法：

```
$ unzip example.docx
Archive:  example.docx
  inflating: [Content_Types].xml
  inflating: _rels/.rels
  inflating: word/_rels/document.xml.rels
  inflating: word/document.xml
  inflating: word/theme/theme1.xml
 extracting: docProps/thumbnail.jpeg
  inflating: word/comments.xml
  inflating: word/settings.xml
  inflating: word/fontTable.xml
  inflating: word/styles.xml
  inflating: word/stylesWithEffects.xml
```

```
  inflating: docProps/app.xml
  inflating: docProps/core.xml
  inflating: word/webSettings.xml
  inflating: word/numbering.xml
$ tree
.
├── [Content_Types].xml
├── _rels
├── docProps
│   ├── app.xml
│   ├── core.xml
│   └── thumbnail.jpeg
└── word
    ├── _rels
    │   └── document.xml.rels
    ├── comments.xml
    ├── document.xml
    ├── fontTable.xml
    ├── numbering.xml
    ├── settings.xml
    ├── styles.xml
    ├── stylesWithEffects.xml
    ├── theme
    │   └── theme1.xml
    └── webSettings.xml
```

正如你所看到的一样，有一些数据结构是由文件和文件夹关系所生成的。其他则在 XML 文件中标注出来。*New Steganographic Techniques for the OOXML File Format, 2011* 论文具体讲述了一些数据隐藏的技巧，但 CTF 竞赛命题人总是会出一些新的想法。

有一个 Python 工具集专门用于查看和分析 OLE 和 OOXML 文档：oletools。对于 OOXML 文档，OfficeDissector 是一个非常强大的分析框架（Python 库）后者还包含了一个快速上手说明。

有时候赛题不一定是找出隐藏的固定数据，也有可能是分析一个 VBA 宏的行为模式，这也是更加贴近现实的一个场景。你可能需要花上一整天来进行这一分析。前文提到的解析工具可以帮助你判断是否存在一个宏，以及如何从中提取出来。一个常见的 VBA 宏是在 Windows 的 Office 文档中，尝试下载一个 Powershell 脚本到 TEMP 文件夹并加以运行，此时你就要开始分析这个 Powershell 脚本。恶意 VBA 宏由于 VBA 相当于一个跨平台代码编译执行框架而易于分析。着案例中你只需要了解复杂的 VBA 宏，以及它的混淆和运行环境，而无须获取微软的许可证进行调试。你也可以使用 Libre Office，它提供了一个友好的调试程序界面，你可以设置端点和生成观察变量，捕获任意时刻运行后的变量值，你也可以使用命令行从特定文件中执行宏：

```
$ soffice path/to/test.docx macro:///.standard.module1.mymacro
```