CTF取证类题目指南

文章简介：大致介绍了CTF比赛中Fornesics题目的常见类型和各类使用工具，此类题目在国内CTF中偶尔出现，但国外的CTF中经常出现这类取证赛题。

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' 00 ff')

>>> example2[1]

255
```

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。

文件雕复

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

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

$ 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？

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。

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

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。

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。

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

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。

抓包（PCAP）文件分析

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。

PDF文件分析

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。

视频与音频隐写

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。

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
```

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