



CTF 之 Forensics 取证

转载

你们这样一点都不可耐  于 2020-04-18 21:04:36 发布  2206  收藏 7

分类专栏: [MISC](#) 文章标签: [安全](#) [经验分享](#) [安全漏洞](#) [加密解密](#) [数据安全](#)

原文链接: <https://www.bodkin.ren/index.php/archives/702/>

版权



[MISC 专栏收录该内容](#)

17 篇文章 0 订阅

订阅专栏

这篇文章列出了CTF中Forensics（取证）类型题的技巧和窍门，展示了CTF中常用工具的使用场景和使用方法。

文件格式（File Formats）

十六进制文件头和对应ASCII码

通过查看文件头前四到五个字节的十六进制数来识别文件类型。参考[Hex file and Regex Cheat Sheet](#) 和 [Gary Kessler File Signature Table](#)

Filetype Start Start ASCII Translation

ani 52 49 46 46 RIFF
au 2E 73 6E 64 snd
bmp 42 4D F8 A9 BM
bmp 42 4D 62 25 BMp%
bmp 42 4D 76 03 BMv
cab 4D 53 43 46 MSCF
dll 4D 5A 90 00 MZ
Excel D0 CF 11 E0
exe 4D 5A 50 00 MZP (inno)
exe 4D 5A 90 00 MZ
flv 46 4C 56 01 FLV
gif 47 49 46 38 39 61 GIF89a
gif 47 49 46 38 37 61 GIF87a
gz 1F 8B 08 08
ico 00 00 01 00
jpeg FF D8 FF E1
jpeg FF D8 FF E0 JFIF
jpeg FF D8 FF FE JFIF
Linux bin 7F 45 4C 46 ELF
png 89 50 4E 47 PNG
msi D0 CF 11 E0
mp3 49 44 33 2E ID3
mp3 49 44 33 03 ID3
OFT 4F 46 54 32 OFT2
PPT D0 CF 11 E0
PDF 25 50 44 46 %PDF
rar 52 61 72 21 Rar!
sfw 43 57 53 06/08 cws
tar 1F 8B 08 00
tgz 1F 9D 90 70
Word D0 CF 11 E0
wmv 30 26 B2 75
zip 50 4B 03 04 PK

SQLite3:

```
0000000: 5351 4c69 7465 2066 6f72 6d61 7420 3300  SQLite format 3.  
0000010: 0400 0101 0040 2020 0000 000b 0000 000b  ...@ ...  
0000020: 0000 0000 0000 0000 0000 0002 0000 0004  ...
```

元数据 (Metadata)

元数据被定义为：描述数据的数据 (data about data)，对数据及信息资源的描述性信息。不同类型的文件具有不同的元数据。照片上的元数据包括日期、相机信息、GPS位置、评论等。音乐上的元数据包括标题、作者、曲目编号和专辑。

时间戳 (Timestamps)

时间戳是指示特定事件(媒体访问控制 MAC)时间的数据：

- 修改（Modification）：当文件被修改时
- 访问（Access）：当文件或条目被读取或访问时
- 创建（Creation）：当文件或条目被创建时

时间戳的类型：

- Modified
- Accessed
- Created
- Date Changed (MFT)
- Filename Date Created (MFT)
- Filename Date Modified (MFT)
- Filename Date Accessed (MFT)
- INDX Entry Date Created
- INDX Entry Date Modified
- INDX Entry Date Accessed
- INDX Entry Date Changed

Timeline Patterns

时间戳伪造？！类似touch命令修改文件的时间属性。

隐写术（Steganography）

Images

如果你在寻找隐藏在图片中的flag，首先要检查：

- **file**, **exiftool**命令来确定文件属性信息。

strings命令，有时候可以通过查看指定长度的字符串来快速判断：

```
strings RainingBlood.mp3 | awk 'length($0)>20' | sort -u
```

- **binwalk**命令，可以确保文件中是否有额外的数据存在。
- **hexdump -C**，展示两列数据，分别为文件的hex值和相应的ASCII码。如果你看到了7z或者PK字样它意味着是压缩文件。如果是这样，你可以使用**7z x**来解压文件。如果你从图片中得到了密码短语，你可以尝试使用strghide工具来解密。
- **stegsolve**图片隐写查看器，检查所有的位图。它有一个数据提取器，我们可以尝试提取RGB不同通道的位值，看看是否有flag。
- **stegosuite**，一款免费并开源的隐写术工具，使用Java编写，你可以很容易的在图像文件中隐藏数据。

steghide，一款隐写术工具，命令行格式，能够将数据隐藏在各种图像和音频文件中。图像文件中的任何文本、图像文件名或任何链接（可能是youtube视频；视频名称可能是密码），都可能是steghide的密码。有时，你也可以尝试它们的大小写组合。

将secret.txt文件隐藏到text.jpg中：

```
steghide embed -cf test.jpg -ef secret.txt -p 123456
```

从text.jpg解出secret.txt：

```
steghide extract -sf test.jpg -p 123456
```

- **zsteg**：在PNG和BMP类型文件中检测隐藏的数据。
- **pngcheck**：pngcheck可以验证PNG，JNG和MNG文件的完整性（通过检查内部32位CRCs [checksum]并解压图像数据）；它可以选择以人类可读的形式转储图像中几乎所有的块级信息。
- **Mediaextract**：提取嵌入到其他文件中的媒体文件（AVI, Ogg, Wave, PNG, ...）。

比较两张相似的图片来发现不同：

```
compare hint.png stego100.png -compose src diff.png
```

- **Image Arithmetic**：我们可以做图像加法，减法，乘法，除法，混合，逻辑与/与非，逻辑或/或非，逻辑异或/异或非，反相/逻辑非，移位运算。

gmic执行图片的异或操作：

```
gmic a.png b.png -blend xor -o result.png
```

- **JPEG**：**Jstego**可将保密信息（如保密文件）隐藏到JPEG图像中，基于Java环境。隐藏算法包含Jsteg和F5。主要的（可能是最难的）东西是JIF文件的编码和解码。
- **JPEG**：**Jsteg**：jsteg可在JPEG文件中隐藏数据，这是一种被称为隐写术的技术。它是将数据的每一位复制到图像的最低有效位（the least-significant bits）来实现的。可隐藏的数据量依赖jpeg文件的大小；它大约需要10-14个字节存储隐藏数据的每个字节。what???
- 修复损坏的JPEG/JPG, GIF, TIFF, BMP, PNG和RAW图像[Repair Corrupted JPEG/JPG, GIF, TIFF, BMP, PNG or RAW Image](#)

LSB 隐写

文件由字节组成，每个字节由8位组成。

```
10101100
```

```
1st digit is MSB and Last digit is LSB
```

改变最低有效位（LSB）并不能使值发生很大的变化。

```
10101100(base 2) == 172 (10)
```

改变LSB从0到1:

```
10101101(base 2) == 173 (10)
```

我们可以修改LSB位数据而不会使文件发生明显的改变，因此可以在这隐藏一些数据。

LSB 图片隐写

LSB隐写术或最低有效位隐写术是一种隐写术方法，其中数据记录在字节的最低位。

假设图像有一个RGB值为（255,255,255）的像素，这个RGB的R通道的值是:

```
1 1 1 1 1 1 1 1
```

我们可以在每个像素的RGB通道上使用其最低位或最低有效位来构造信息:

```
1 1 1 1 1 1 1 0
```

由于1bit改变带来的颜色差异不大，导致隐写后肉眼难以发现区别:

Color 1	Color 2
FFFFFFE	FFFFFF

解码LSB隐写与编码原理完全相同，但行为相反。大致过程是获取LSB数值并记录，当搜集完每个LSB位值后，再将其转换成文本或文件。

二维码?

下载zbarimg:

```
apt-get install zbar-tools
```

读取二维码:

```
zbarimg <imagefile>
```

得到一个二进制0101...的二维码，使用[QR Code Generator](#)转换它。

Sound Files

使用**Audacity**打开文件，分析频谱图，参考[Spectrum Analyzer](#)。

- 点击轨道旁边的箭头，从波形图（顶部）切换到频谱图（底部）。
- 可能是莫尔斯电码。由于所有莫尔斯数据都低于100赫兹，我们可以使用低通滤波器（effects menu, cutoff 100 Hz），方便查看数据。
- [Golang mp3 Frame Parser](#)

例子:

如果你发现如下规律的频谱:

它也许存在二进制数据:

结果可能是:

```
11111110 11111110
01010110 00010101
```

PCAP

- **Wireshark**, 在pcap文件中寻找目标?

在wireshark中搜索HTTP Web流量中泄露的密文:

```
http.request.method == "POST" filter might help, based on concept that server is asking for LOGIN prompt and user is POSTing his password in cleartext.
```

- 可以使用"&&"符号将过滤器链接在一起, 请确保使用双等于"="。
- 如果ip地址已经被欺骗, 那么你应该寻找MAC地址, 因为它不会变。您也许会发现两个不同ip地址的数据包具有相同的MAC地址。另一种情况, 如果MAC地址被欺骗, 则ip地址也可能是相同的。面对这两种情况, 过滤"arp" (仅显示arp请求) 和"ip.addr" (仅显示源或目的地ip地址的数据包) 可能会有所帮助。
- 有时, 最好检查下我们能够导出哪些对象。(File -> Export Objects -> HTTP/DICOM/SMB/SMB2) 导出http/DICOM/SMB/SMB2对象。
- SSL流量? 有key吗? 有的话设置key: Wireshark->Edit->Preferences->Protocols->SSL->RSA Key List, 重新加载wireshark可以解密流量。

有时, 你需要在流量包中找到所有ip地址:

```
tshark -T fields -e ip.src -r <pcap file> | sort | uniq
```

-T fields|pdm|ps|psml|text : Set the format of the output when viewing decoded packet data.

-e : Add a field to the list of fields to display if -T fields is selected.

-r : Read packet data from infile, can be any supported capture file format (including gzipped files).

-R : Cause the specified filter (which uses the syntax of read/displayfilters, rather than that of capture filters) to be applied

- Wireshark不能直接分配http碎片来生成RAW数据, 我们可以使用Dshell来重组HTTP部分内容, 这里有一个[博客](#)提到如何做到这一点。...跟踪流???
- 如果PCAP文件包含有其他文件, 可以尝试使用binwalk或foremost来提取文件。

• USB Forensics

一个基于USB流量的PCAP文件, 可能是USB-鼠标、键盘、存储设备的流量。拿到数据后首先来看看USB连接了什么设备。使用wireshark检查数据包:

```
1 0.000000 host 1.12.0 USB 36 GET_DESCRIPTOR Request DEVICE
```

```
2 0.000306 1.12.0 host USB 46 GET_DESCRIPTOR Response DEVICE
```

在**GET_DESCRIPTOR**返回包中，有**idVendor**和**idProduct**字段，根据此数据我们可以知道它是键盘、鼠标还是存储设备。

```
DEVICE_DESCRIPTOR

bLength: 18

bDescriptorType: 0x01 (DEVICE)

bcdUSB: 0x0200

bDeviceClass: Device (0x00)

bDeviceSubClass: 0

bDeviceProtocol: 0 (Use class code info from Interface Descriptors)

bMaxPacketSize0: 8

idVendor: Razer USA, Ltd (0x1532)

idProduct: BlackWidow Ultimate 2013 (0x011a)

bcdDevice: 0x0200

iManufacturer: 1

iProduct: 2

iSerialNumber: 0

bNumConfigurations: 1
```

USB-Keyboard

如果设备连接的是键盘，我们可以检查**interrupt in**消息。

```
51 8.808610 1.12.1 host USB 35 URB_INTERRUPT in
```

检查 the Leftover Capture Data field:

```
Frame 159: 35 bytes on wire (280 bits), 35 bytes captured (280 bits)

USB URB

[Source: 1.12.1]

[Destination: host]

USBPcap pseudoheader length: 27

IRP ID: 0xfffffa5045d1653c0

IRP USBD_STATUS: USBD_STATUS_SUCCESS (0x00000000)

URB Function: URB_FUNCTION_BULK_OR_INTERRUPT_TRANSFER (0x0009)
```

```
IRP information: 0x01, Direction: PDO -> FDO

URB bus id: 1

Device address: 12

Endpoint: 0x81, Direction: IN

URB transfer type: URB_INTERRUPT (0x01)

Packet Data Length: 8

[bInterfaceClass: HID (0x03)]

Leftover Capture Data: 0000500000000000
```

我们使用tshark来取出数据，保存为usb.capdata。

```
tshark -r usb-keyboard-data.pcap -T fields -e usb.capdata

00:00:08:00:00:00:00:00
00:00:00:00:00:00:00:00
00:00:0e:00:00:00:00:00
00:00:00:00:00:00:00:00
00:00:16:00:00:00:00:00
```

每一数据有8个字节。

Keyboard Report Format

- Byte 0: Keyboard modifier bits (SHIFT, ALT, CTRL etc)
- Byte 1: reserved
- Byte 2-7: Up to six keyboard usage indexes representing the keys that are currently “pressed”. Order is not important, a key is either pressed (present in the buffer) or not pressed.

键盘发送 02 00 0e 00 00 00 00 00，表示同时按下了Left Shift + k，即大写K。

具体键位对应参见Hut1_12v2.pdf第53页。

USB HID Keyboard Scan Codes

MightyPork根据USB规范1.11编写了一个USB HID键盘扫描码，记录在[usb_hid_keys.h](#)。

可参考以上代码内容编写脚本，转换usb.capdata的数据来查看用户的键盘使用记录！

whoami已经写了一个Python脚本：

```
usb_codes = {

0x04:“aA”, 0x05:“bB”, 0x06:“cC”, 0x07:“dD”, 0x08:“eE”, 0x09:“fF”,

0x0A:“gG”, 0x0B:“hH”, 0x0C:“iI”, 0x0D:“jJ”, 0x0E:“kK”, 0x0F:“lL”,

0x10:“mM”, 0x11:“nN”, 0x12:“oO”, 0x13:“pP”, 0x14:“qQ”, 0x15:“rR”,
```



```

0x16:"sS", 0x17:"tT", 0x18:"uU", 0x19:"vV", 0x1A:"wW", 0x1B:"xX",
0x1C:"yY", 0x1D:"zZ", 0x1E:"1!", 0x1F:"2@", 0x20:"3#", 0x21:"4$",
0x22:"5%", 0x23:"6^", 0x24:"7&", 0x25:"8*", 0x26:"9(", 0x27:"0)",
0x2C:" ", 0x2D:"-_", 0x2E:"=+", 0x2F:"[{", 0x30:"]}", 0x32:"#~",
0x33:";:", 0x34:"'\"", 0x36:",<", 0x37:".>", 0x4f:">", 0x50:"<"
}

lines = ["", "", "", "", ""]

```

```

pos = 0
for x in open("data1.txt", "r").readlines():
    code = int(x[6:8], 16)

    if code == 0:
        continue

```

newline or down arrow - move down

```

if code == 0x51 or code == 0x28:
    pos += 1
    continue

```

up arrow - move up

```

if code == 0x52:
    pos -= 1
    continue

```

select the character based on the Shift key

```

if int(x[0:2], 16) == 2:
    lines[pos] += usb_codes[code][4]
else:
    lines[pos] += usb_codes[code][0]

for x in lines:
    print x

```

USB-Mouse

如果我们捕获USB鼠标的流量数据，可以发现记录数据使用了四个字节。鼠标移动时表现为连续性，与键盘击键的离散性不一样，不过实际上鼠标动作所产生的数据包也是离散的。

第一个字节有一堆flag标志，它代表鼠标按键。0代表未按键，1代表左键，2代表右键，其余字段看意思。

```

byte 1:
Y overflow   X overflow   Y sign bit   X sign bit   Always 1   Middle Btn   Right Btn
Left Btn

```

第二个字节是“X”的值，它代表鼠标在水平方向的移动，向左为负。

```
byte 2:
```

```
X movement
```

第三个字节是“Y”的值，它代表鼠标在垂直方向的移动，向下（朝向用户）为负。

```
byte 3:
```

```
Y movement
```

第四个字节是扩展字节，当鼠标有滚轮的时候才会被激活。

假设我们已经将这些数据捕获到一个文件中，我们可以从中直接提取鼠标移动的数据，

```
tshark -r challenge.pcapng usb.capdata and usb.device_address12 -T fields -e usb.capdata > mouse_data.txt
```

这里可以使用[GNUplot](#)来绘制，参考 [Riverside](#)

```
awk -F: 'function comp(v){if(v>127)v-=256;return v}{x+=comp(strtonum("0x"$2));y+=comp(strtonum("0x"$3))}$1"01"{print x,y}' mouse_data.txt > click_coordinates.txt
```

GNUplot

```
gnuplot -e "plot 'click_coordinates.txt'"
```

如果鼠标移动在屏幕键盘上，我们可以使用

```
awk 'BEGIN{split("zxcvbnm asdfghjkl qwertyuiop",key,/,/)}{r=int(($2-20)/-100);c=int(117+/, file disk*
```

```
$ ls -lh
```

```
512K disk0
```

```
12 disk1
```

```
512K disk2
```

```
$ cat disk1
```

```
crashed □)
```

从结果可以看出disk1损坏了。但也看出了这里使用了RAID。RAID允许如果3个磁盘丢失1个，通过异或其他两个磁盘可以获得每个磁盘的版本。我们使用python来异或disk0和disk2来获取disk1：

```
from pwn import *

with open("disk0", "rb") as f1:

with open("disk2", "rb") as f2:

with open("disk1", "wb") as f3:

x = f1.read()

y = f2.read()

f3.write(xor(x,y))
```

或者可以使用[xor-files](#)对两个或多个文件进行异或。

现在，为了得到完整的NAS内容，我们必须确定数据块的分布。在分析了磁盘内容，了解了一些FAT12结构后，我们已经

确定奇偶校验块（BP）位于每行的不同磁盘上，因此我们有：

```
D0 | D1 | D2
-|---|-
B0 | B1 | BP
B2 | BP | B3
BP | B4 | B5
B6 | B7 | BP
```

使用python将所有的数据块拼凑起来：

```
n = 1024
k = 512 # block size

with open("disk0", "rb") as f1:
with open("disk1", "rb") as f2:
with open("disk2", "rb") as f3:
with open("disk_out", "wb") as f_out:
x = 2
for _ in xrange(n):
blocks = (f1.read(k), f2.read(k), f3.read(k))
data_blocks = [b for i, b in enumerate(blocks) if i != x]
x = (x - 1) % 3
f_out.write("".join(data_blocks))
```

现在我们可以挂载新生成磁盘来检查内容。

格式（Formats）

Boarding Pass Format

机场签发的登机牌 from [What's contained in a boarding pass barcode?](#)

```
M1EWING/SHAUN          E1AAAAA SYDBNEQF 0524 106Y023A0073 359>2180
B                      29          0      QF 1245678          128
```

登机牌条形码上有很多信息，解释如下：

- M1 : Format code 'M' and 1 leg on the boarding pass.
- EWING/SHAUN : My name.
- E1AAAAA : Electronic ticket indicator and my booking reference.
- SYDBNEQF : Flying from SYD (Sydney) to BNE (Brisbane) on QF (Qantas).
- 0524 : Flight number 524.
- 106 : The Julian date. In this case 106 is April 16.
- Y : Cabin – Economy in this case. Others including F (First) and J (Business).
- 23A : My seat.

- 0073 : My sequence number. In this case I was the 73rd person to check-in.
- 3 : My “passenger status”.
- 59 : There is a various size field. This is the size
- > : Beginning of the version number
- 2 : The version number.
- 18 : Field size of another variable field.
- 0 : My check-in source.
- B : Airline designator of boarding pass issuer.
- 2 : Another variable size field.
- 9 : Airline code.
- 0 : International document verification. '0' as I presume is not applicable.
- QF : The airline my frequent flyer account is with.
- 1245678 : My frequent flyer number.
- 128 : Airline specific data.

Interesting Blog

- [APT-Incident-Response](#)
- [Securityfest CTF - Coresec challenge writeup](#)
- [SHX7 - for300](#)

Others

- [Unicode](#)

安卓逆向，下面提供三种反编译方法：

Apktool: 提取所有资源。apktool也可将apk文件转换为smali格式。

```
apktool d file.apk output-dir
d : decode to output-dir
```

Dex2jar: 查看java代码

- 将apk文件后缀改成zip
- 解压zip，得到classes.dex
- 使用dex2jar classes.dex
- 通过jar xf classes_dex2jar.jar 提取jar文件
- 使用jd-gui查看反编译后的java代码
- 使用Decompile Android等在线服务。一旦被反编译，我们就可以下载反编译后的文件并审计它。

IOS包，使用dpkg-deb来提取：

```
dpkg-deb -x com.yourcompany.whyo_4.2.0-28debug_iphoneos-arm.deb app
```

- disk...img文件，使用foremost提取，或修复文件头，文件尾，数据结构。
- jar文件：

```
jar xf jar-file
```

x : extract files from the JAR archive.

f : JAR file from which files are to be extracted is specified on the command line, rather than through stdin.

The jar-file argument is the filename (or path and filename) of the JAR file from which to extract files

- 如果是恶意代码文件，那么在真实的环境下执行做下比较来定位恶意代码。
- 摩尔斯代码，利用[Transator](#)。

有时候提取一些文件，会看到一些空名字文件：

```
ls -lb might be of help.
```

```
-b, --escape : print C-style escapes for nongraphic characters
```

打开一个文件名为 `-` 的文件：

```
cat ./-
```

- Excel文档，可以尝试解压它，并检查其中的VBA宏。

GIF to JPG

```
convert animation.gif target.png
```

- 这一块像是作者随手的笔记...?!

转自:<https://www.bodkin.ren/index.php/archives/702/>

作者:SewellDinG 老锥