Data Extraction

EN | ZH 这一块是流量包中另一个重点, 通过对协议分析, 找到了题目的关键点, 如何提取数据成了接下来的关键问题

wireshark

wireshark 自动分析

`file -> export objects -> http`

手动数据提取

`file->export selected Packet Bytes`

tshark

tshark 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

再看`Google CTF 2016 Forensic-200`这一题, 可以通过 tshark 迅速完成解题

```
what@kali:/tmp$ tshark -r capture.pcapng -T fields -e usb.capdata >
data2.txt
what@kali:/tmp$ # awk -F: 'function comp(v){if(v>127)v-=256;return v}
{x+=comp(strtonum("0x"$2));y+=comp(strtonum("0x"$3))}$1=="01"{print x,y}'
data.txt > data3.txt
what@kali:/tmp$ gnuplot
> plot "data3.txt"
```

- Step 1 鼠标协议中数据提取
- Step 2 通过 awk 进行位置坐标转换
- Step 3 形成图形

常用方法

> `tshark -r **.pcap -Y ** -T fields -e ** | **** > data`

```
Usage:
  -Y <display filter>      packet displaY filter in Wireshark display
filter
                           syntax
  -T pdml|ps|psml|json|jsonraw|ek|tabs|text|fields|?
                           format of text output (def: text)
  -e <field>               field to print if -Tfields selected (e.g.
tcp.port,
                           _ws.col.Info)
```

通过 `-Y` 过滤器 (与 wireshark 一致), 然后用 `-T filds -e` 配合指定显示的数据段 (比如 usb.capdata)

- `tips`
  - `-e` 后的参数不确定可以由 `wireshark` 右击需要的数据选中后得到

例题

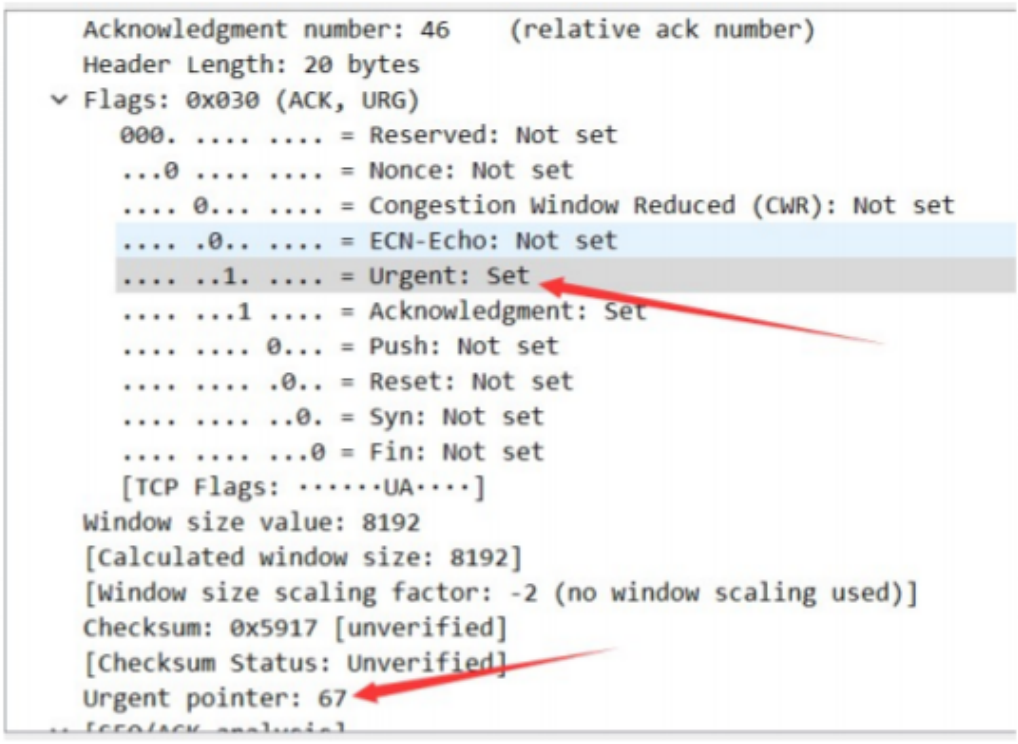
> 题目：`google-ctf-2016 : a-cute-stegosaurus-100`

这题的数据隐藏的非常巧妙, 而且有一张图片混淆视听, 需要对 `tcp` 协议非常熟悉, 所以当时做出来的人并不多, 全球只有 `26` 支队伍

在 `tcp` 报文段中有 6Bit 的状态控制码, 分别如下

- URG：紧急比特（urgent）, 当 URG = 1 时，表明紧急指针字段有效, 代表该封包为紧急封包。它告诉系统此报文段中有紧急数据，应尽快传送 (相当于高优先级的数据)
- ACK：确认比特（Acknowledge）。只有当 ACK = 1 时确认号字段才有效, 代表这个封包为确认封包。当 ACK = 0 时，确认号无效。
- PSH：（Push function）若为 1 时，代表要求对方立即传送缓冲区内的其他对应封包，而无需等缓冲满了才送。
- RST：复位比特 (Reset) , 当 RST = 1 时，表明 TCP 连接中出现严重差错（如由于主机崩溃或其他原因），必须释放连接，然后再重新建立运输连接。
- SYN：同步比特 (Synchronous)，SYN 置为 1，就表示这是一个连接请求或连接接受报文, 通常带有 SYN 标志的封包表示『主动』要连接到对方的意思。。
- FIN：终止比特 (Final)，用来释放一个连接。当 FIN = 1 时，表明此报文段的发送端的数据已发送完毕，并要求释放运输连接。

而这里的 `tcp.urg` 却为



通过 tshark 提取 `tcp.urg` 然后去除 0 的字段, 换行符转 , 直接转换成 python 的列表, 转 ascii 即可得到 flag

```
root@kali:  tshark -r Stego-200_urg.pcap -T fields -e
tcp.urgent_pointer|egrep -vi "^0$"|tr '\n' ','
Running as user "root" and group "root". This could be dangerous.
67,84,70,123,65,110,100,95,89,111,117,95,84,104,111,117,103,104,116,95,73,116,

...
>>> print "".join([chr(x) for x in arr]) #python转换ascii
CTF{And_You_Thought_It_Was_In_The_Picture}
```

> 题目: `stego-150_ears.xz`

Step 1

通过 `file` 命令不断解压得到 `pcap` 文件

```
➜  Desktop file ears
ears: XZ compressed data
➜  Desktop unxz < ears > file_1
➜  Desktop file file_1
file_1: POSIX tar archive
➜  Desktop 7z x file_1

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,1 CPU
Intel(R) Core(TM) i7-4710MQ CPU @ 2.50GHz (306C3),ASM,AES-NI)
```

```
Scanning the drive for archives:
1 file, 4263936 bytes (4164 KiB)

Extracting archive: file_1
--
Path = file_1
Type = tar
Physical Size = 4263936
Headers Size = 1536
Code Page = UTF-8

Everything is Ok

Size:       4262272
Compressed: 4263936
```

Step 2

通过 `wireshark` 发现 `dns` 中回应名字存在异常，组成 `16` 进制的 `png` 文件

采用 `tshark` 进行提取，提取 `dns` 中的数据, 筛选具体报文形式 `\w{4,}.asis.io`

```
tshark -r forensic_175_d78a42edc01c9104653776f16813d9e5 -T fields -e dns.qry.name -e dns.flags|grep 8180|awk '{if ($1~/\w{4,}.asis.io/) print $1}'|awk -F '.' '{print $1}'|tr -d '\n' > png
```

Step 3

`16` 进制还原图片

```
xxd -p -r png flag
```

自定义协议

提取数据存在一类特殊情况，即传输的数据本身使用自定义协议，下面用 `HITCON 2018` 的两道 Misc 为例说明。

例题分析

- HITCON-2018 : ev3 basic
- HITCON-2018 : ev3 scanner

ev3 basic

确定数据

对于这类题目，首先分析有效数据位于哪些包中。观察流量，通讯双方为 `localhost` 和 `LegoSystem` 。其中大量标为 `PKTLOG` 的数据包都是日志，此题中不需关注。简单浏览其余各

个协议的流量，发现仅 `RFCOMM` 协议中存在没有被 `wireshark` 解析的 `data` 段，而 `RFCOMM` 正是蓝牙使用的传输层协议之一。

由前述 `tshark` 相关介绍，可以通过以下命令提取数据：

```
tshark -r .\ev3_basic.pklg -T fields -e data -Y "btrfcomm"
```

分析协议

找到数据后，需要确定数据格式。如何查找资料可以参考 `信息搜集技术` 一节，此处不再赘述。总之由 `ev3` 这个关键词出发，我们最终知道这种通信方式传输的内容被称之为 Direct Command，所使用的是乐高自定义的一种简单应用层协议，`Command` 本身格式由乐高的手册 EV3 Firmware Developer Kit 定义。（查找过程并不像此处简单而直观，也是本题的关键点之一。）

在乐高的协议中，发送和回复遵从不同格式。在 `ev3 basic` 中，所有回复流量都相同，通过手册可知内容代表 `ok` ，没有实际含义，而发送的每个数据包都包含了一条指令。由协议格式解析出指令的 `Opcode` 均为 `0x84` ，代表 `UI_DRAW` 函数，且 `CMD` 是 `0x05` ，代表 `TEXT` 。之后是四个参数，`Color`，`X0`，`Y0`，`STRING` 。此处需要注意乐高的单个参数字节数并不固定，即便手册上标明了数据类型是 `DATA16` ，仍然可能使用一个字节长度的参数，需要参照手册中 `Parameter encoding` 一节及相关文章。

尝试分析几个命令，发现每个指令都会在屏幕特定位置打印一个字符，这与提供的图片相符。

处理结果

理解数据内容后，通过脚本提取所有命令并解析参数，需要注意单个参数的字节数不固定。

得到所有命令的参数后，可以将每个字符其按照坐标绘制在屏幕上。较简单的做法是先按 `X` 后按 `Y` 排序，直接输出即可。

ev3 scanner

第二题的做法与第一题基本相同，难度增加的地方在于：

- 发送的命令不再单一，包括读取传感器信息、控制 ev3 运动
- 回复也包含信息，主要是传感器读取的内容
- 函数的参数更复杂，解析难度更大
- 解析命令得到的结果需要更多处理

`ev3 scanner` 此处不再提供详细方法，可作为练习加深对这一类型题目的理解。

Python Script

TODO