网络工具（二）

在纯终端环境下，我们需要对网络进行故障排查或调试时，最终极的方式就是能够完整读取收发数据的所有内容。我们需要一款网络嗅探器，能够获取网络中的信息包，在Linux下就是tcpdump命令。

抓包三要素我觉得是抓取、筛选和阅读。其中前两部分我们将结合tcpdump提供的基本选项和过滤语法来进行。而第三部分对于抓取的数据进行解读的过程则需要我们对TCP、UDP、IP乃至ICMP报文头的各个字段的功能和意义都非常熟悉才行，而且我们在网卡处于混杂模式下嗅探到的数据包会非常多，只有充分了解我们所抓去包的性质才能过滤掉无意义的数据包。

首先我们可以通过命令tcpdump –D来查询当前所有可以选择的抓包对象。之后我们可以从这些对象中选出我们需要监听的对象（网卡）。

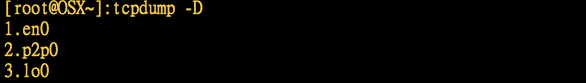


图1：可监听对象

接下来我们可以从这三个网卡中选择一个用于监听的对象，这里我们选择以太网网卡（en0）作为指定的网络接口。具体的操作就是在tcpdump之后加上选项-i en0。这时如果开始执行命令，嗅探器就将会监听以太网网卡接口所有的数据。

除了确定监听的接口之外，我们还可以设置通过以下选项来控制输出的内容。

-nn 当设置nn选项之后，tcpdump遇到周知协议明或者端口号时直接显示数字而不将其转换成协议名或端口名。

-X需要把协议头和包内容原原本本地显示出来，会同时以十六进制和ASCII形式显示。

-c 这里c是count的意思，表示抓取的数据包的数量。

-e 将会输出以太网头的信息，比如可以查看到数据包的源和目的mac地址。

-t 在不加t选项的情况下，会打印出每个包被抓取的时间，加上后该项会省去。

-F 通常我们仔进行筛选时表达式都会比较长，所以我们可以把筛选表达式存储为独立的文件，然后通过-F命令读取过滤表达式，使用方式是-F filename。

-w 我们可以将当前抓去的包的信息存储起来，记录成为文件形式，这样可以在以后调出查询。这里的功能就相当于日志一般。

-r 和-w命令配套，用于读取通过-w保存的数据包的信息，当读取了该文件之后，参数行的选项和过滤条件一就可以被使用，其操作和及时抓取的数据是一样的。

接下来我们将通过一个具体的例子展示上面各个选项功能，这里将会一次性使用多个参数。有一点需要说明的是，当我们键入指令后如果计算机包括所在局域网内没有任何数据传输活动的话，那么我们是不会抓去到任何信息的，所以萎了保证立马可以抓去到内容，我们可以仔自己浏览器下输入一个可以连通的网站并且访问，这样我们就可以抓取到通过以太网网卡的数据了。

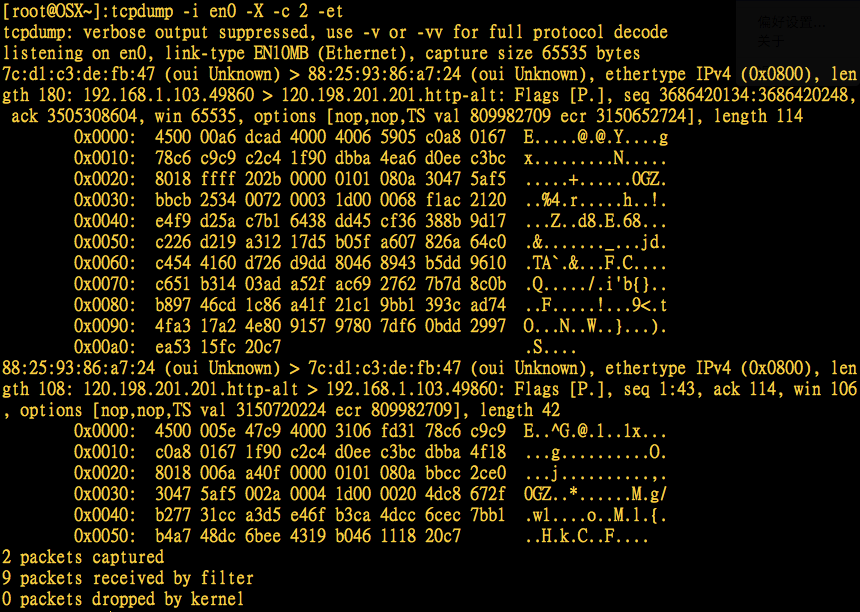


图2：tcpdump抓包示例

对比前面的描述，这里从第三行listening on en0可以看出tcpdump只监听的是以太网网卡的数据，共抓去了两个数据包并且显示了mac地址信息位显示时间与-cet选项对应，而显示的数据包含协议头和文件的十六进制内容与-X选项对应。

我们接下来输入tcpdump –X –w tcpfile，这样可以把当前抓到的包保存到文件tcpfile中。但是这个文件是通过raw packet（原始网络包）形式存储下来的，存储的都是结构体数据，无法直接通过cat或者less打开。不过我们可以通过-r命令来读取这个文件的内容。

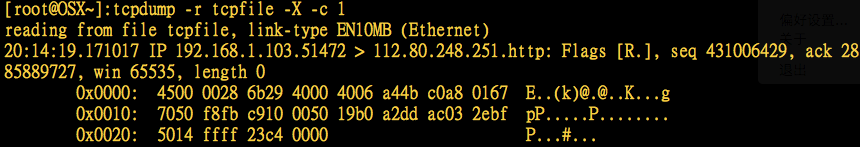


图3 ：回放抓取数据

这里通过-r读取了之前通过-w命令保存网络数据包，在保存tcpfile的时候我们没有附加什么限制，不过在读取该文件的时候我们可以如同及时操作一般控制输出的内容。比如在这里使用-c 1控制只输出第一条被抓取数据的信息（原本文件中有多条信息），-X则是为了完整输出信息。这里需要注意的一个地方是第三行开始是抓包时间，这和我当前时间不同，也证明了该文件是之前抓包信息的一个保存。

上面简单介绍了tcpdump的几个重要选项，但是还有一个问题必须解决，就是我们通过监听以太网网卡可以的到的数据实在太多，所以我们必须从这些数据中筛选出符合我们需要数据包，否则抓再多都不能帮助解决实际问题。

数据包的过滤我们可以从以下几点来考虑：

1. 特定IP地址/主机（host）
2. 使用特定协议（udp/tcp等）
3. 特定端口号（port）
4. 特定网段（net）
5. 特定字段信息

以上前四个会比较容易理并实现，我们只需要指定特定过滤信息即可。我们过滤的实现是定义在参数单引号之间的，如果所需的过滤条件不止一条的话，还可以通过and、or、not来连接多个条件，这和些一般程序中的逻辑判断并无二异。

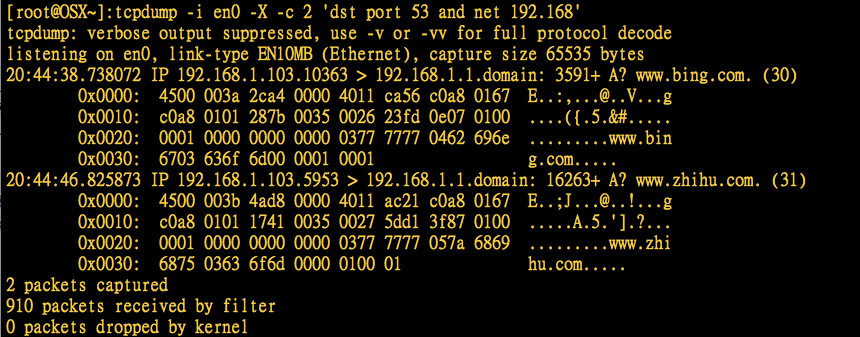


图4：过滤示例

选项部分的内容不再重复，我们从过滤条件dst port 53 and net 192.168来入手。在指定过滤条件的时候，除了指定网络或者端口之外还可以通过dst和src来指定是目的地址还是源地址。这里就是指定目的地址中端口号为53且网段为192.168的数据包（这里这么筛选的原因是因为配置的本地DNS服务器地址为192.168.1.1，且DNS服务器监听的端口为53，所以一旦在浏览器输入网址涉及DNS时就可以抓取相应的数据包）。

这里port之后还可以写成是dns，在配置文件/etc/services中保存了其中的对应关系，比如ftp-data对应20，ftp对应21，telnet对应23。小于1024的端口号为周知端口号，由IANA全权负责。

通过特定字段筛选的内容需要我们先能够读懂TCP、UDP、IP协议头各字段的功能。所以我们接下的工作将分成三部分：

1. 简要介绍协议头字段信息
2. 抓包并结合上面知识分析
3. 通过特定字段信息过滤数据

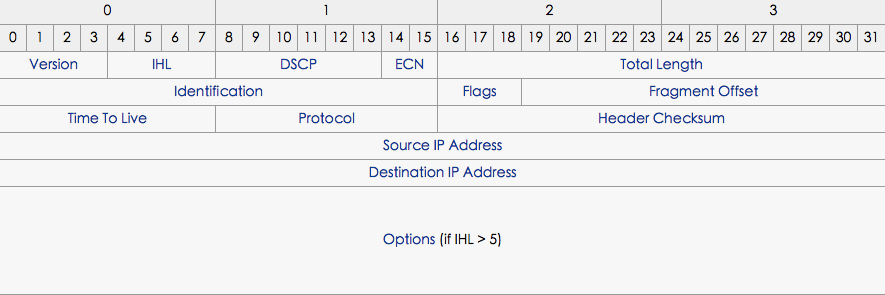


图5：IP报文头

Version（版本）：表示IP协议版本，IPv4协议此段为4。

IHL（报文头长）：以4字节为单位，只表示报文头长度。

DSCP + ECN（TOS）：暂时不去考虑，通常为0。

Total Length（总长度）：IP头加其数据长度。

Identification（标识域）：通过链路层时会因为链路能承载的最大传输单元（MTU）大小限制需要将数据报分片，并在目的端重新通过此标识来判断多个分片原本是不是属于同一个数据报。

Flags：三个bit位分别是[R, DF, MF]，其中R位为保留位，DF位为是否允许分片的，为1时表示不允许分片，此时第三个MF位就失效了。如果数据报超过了链路层MTU大小，且DF为1的话，那么该数据报将会被丢弃。如果DF为0且数据报被分片，每个数据报在目的结点就可以通过MF为是否为1来判断是否是最后一个分片，最后一个分片MF位为0。

Fragment Offset（片偏移）：以8字节为单位，表示该IP数据报数据部分第一个字节在原数据中的偏移量。（注意，这里是数据的偏移量。要先去除掉IP头的20字节）

TTL（生命周期）：每路过一个路由器减1，为0时丢弃。

Protocal（协议）：传输层使用的协议，17为UDP，6为TCP。

Header Checksum（首部校验和）：校验IP首部信息是否正确。

Source IP（源IP）：源IP地址。

Destination IP（目的IP）：目的IP地址。

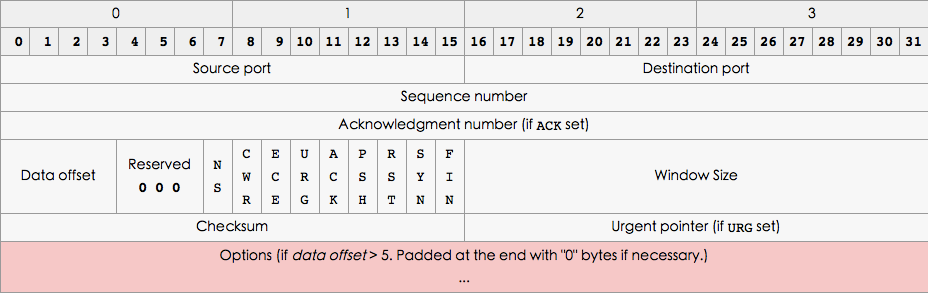


图6：TCP报文头

Source port（源端口）：源端口号。

Destination port（目的端口）：目的端口号。

Sequence number（序号）：所发送数据块在整个数据中的字节编号。

Acknowledgement number（确认号）：反馈给对方的信息，通常为对方发送数据seq的值加上发送数据的大小，表示请求下一个数据块第一个字节的编号。

Data offset（数据偏移）：实际就是TCP头部长度。

Reserved（保留）：未使用。

Tag（标志字段）：这里简单介绍下。URG：紧急指针有效位，ACK：确认序号有效位（为1时ACK NUM有效），PSH：接收方应尽快交给应用层，RST：重建连接，SYN：序号有效位（为1时SEQ NUM有效），FIN：释放连接。

Window size（窗口大小）：用来进行流量控制，字节为单位，是本机期望接收的字节数。

Checksum（校验和）：尤其注意这里的校验和不再是IP层的首部校验和，这里因为传输层是直接与应用层相邻的层，所以要连同数据一同进行校验。

Urgent pointer（紧急指针）：是一个偏移量，和seq字段中的值相加表示紧急数据最后一个字节的序号。

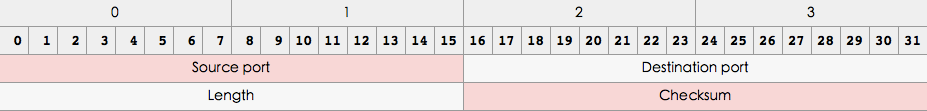


图7：UDP报文头

其功能算是TCP子集了，不再逐一描述字段信息。

接下来抓个包分析。

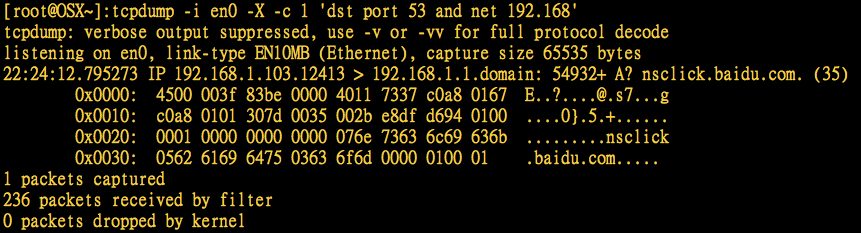


图8：抓包示例

其实看请求行就可以知道我们抓去的是一个DNS请求的包。

从宏观上看是本机（IP地址为192.168.1.103）通过12413端口号发送请求到本地DNS服务器（IP地址为192.168.1.1）的53号DNS服务器监听端口请求域名解析服务，我们所请求的内容从第四行大致可以看出来是在请求baidu.com的A记录。

从地址0x0000开始就是我们抓取到的数据包信息了，我们先不去看后面ASCII信息。前半部分都是十六进制的数据，所以每两个数字表示一个字节，比如最开始的0x4500，其中0x4和0x5组成第一个字节数据，0x0和0x0组成第二个字节。

还有一点需要说明的是，我们抓去到的是IP数据报，也就是说从第一个字节开始就是IP头的内容，通常IP头去除掉之后就是其有效数据（一般由TCP/UDP头加上其有效数据组成）。所以我们要做的就是从第一个字节其逐个字节解析IP头内容，根据给出的IP头大小定位传输层头部的内容，最后再去分析实际传输的数据。

第一个字节拆成0x4和0x5，对应IP头的信息表示，4表示使用的是IPv4协议，接下来的5是用来表示头部大小的，单位是4字节，所以就确定了这个报文的IP头大小为4 \* 5 = 20字节，也就是最常规的情况。接下来TOS字段的一个字节数据跳过。0x003f指代的是IP数据报的大小，转换成10进制就是（1+2+4+8+16+32 = 63字节），实际上我们去数这个数据报大小时也能得到这个结果（每行16字节，数据报占了4行缺一个字节）。0x83be这两个字节数据表示标识域，用来组装数据。0x0000这两个字节全为0说明我们这个数据报没有被分片，自然也不存在数据偏移量。0x40表示的TTL大小，换算成十进制就是64，也就是说这个DNS请求数据报被发送到网络中，如果经过了64个路由器的话就会自动被丢弃。接下来的0x11换算成十进制代表数字17，对应的上层服务为UDP服务，因为我们这里是DNS请求报文，而DNS默认使用的传输层协议就是UDP，所以这里和我们的认知还是相符的。接下来0x7337是首部校验和，但一定要记住在IP层我们只需要校验首部，而在传输层我们需要检验的是首部及数据。0xc0a8 0167->(0xc0:0xa8:0x01:0x67)->(192.168.1.103)表示源IP地址，0xc0a8 0101->(0xc0:0xa8:0x01:0x01)->(192.168.1.1)表示目的主机IP地址。

在分析IP头部时候有两个字段对我们接下来的工作非常重要，一个是IP头部大小是20字节，另一个是上层使用的协议是UDP协议。因为IP头部大小是20字节，所以从下一个字节开始将会是传输层UDP协议的首部，而UDP头的大小只有8个字节，所以我们很快就可以完成分析。

UDP头部很特殊，每一个有效字段在读取时候都是按照两字节来读的。UDP头的前两个字节数据为0x30d7转换成十进制就是我们之前提到的客户机端口号12413，接下来两个字节是目的主机端口号0x0035，转换成十进制正好是DNS的周知端口号53。再接下来两个字节是0x002b，十进制大小为43，一开始就提到整个IP数据报的大小为63，IP头的大小为20，所以这里43正好是UDP头部加数据的大小，往前想一步，这里UDP头的大小为8字节，所以我们可以推知传输层传递给应用层的数据大小为35字节。最后两个字节0xe8df是校验和的值，这里校验的是头部和整个数据部分。

当数据通过传输层之后，将会被摘去这8字节的UDP头部，上传给应用层的数据将只有35字节。此时数据将会被从内核态的TCP/IP程序传给用户态的DNS解析程序。对于DNS协议各个字段的含义暂时我并没有去研究，但是可以从右侧ASCII码打印的部分值来看，这里包过的数据里至少包含我们在请求DNS域名解析所需要的域名字符串baidu.com。