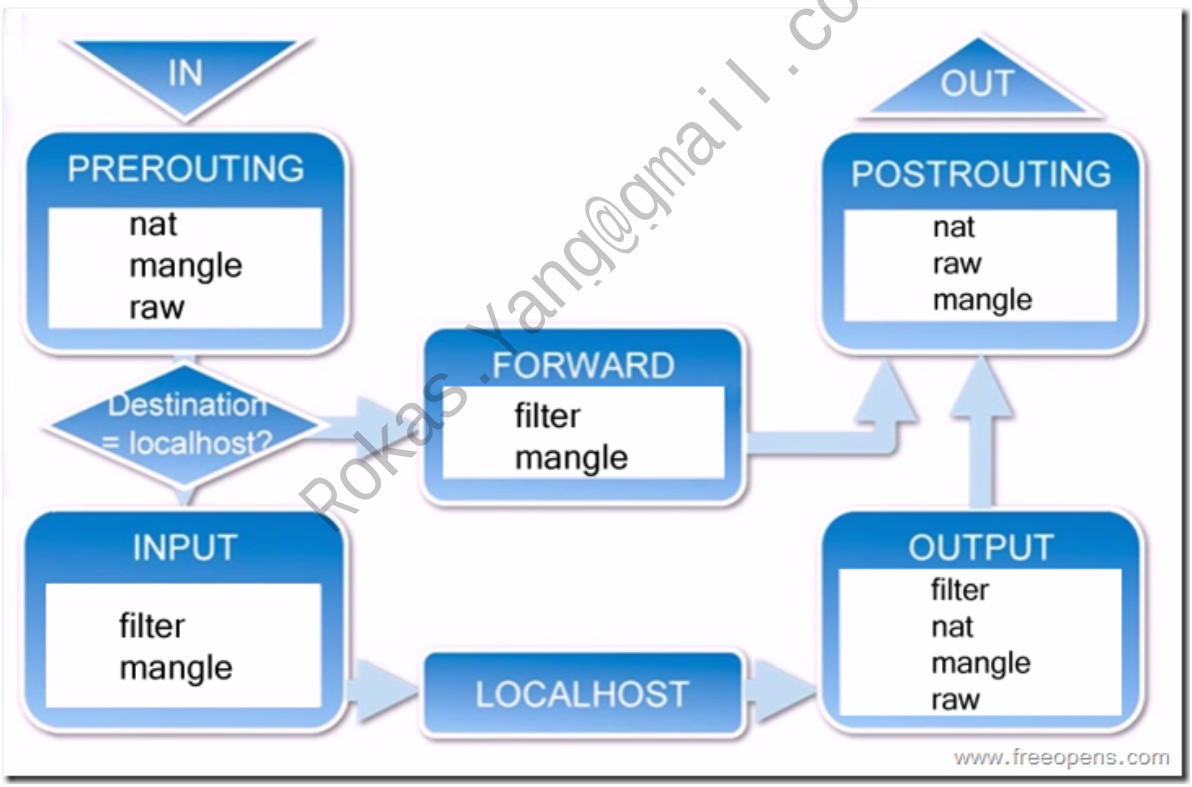
抓包神器TCPDUMP的分析总结-涵盖各种使用场景、高级用法

一、前言

网络故障排查中，经常要抓包，windows有wireshark，linux最常用的是tcpdump，其中被问得最多的一个问题："iptables限制后，tcpdump还能抓到包吗？",首先看下数据包进入OS及出去的顺序：

`网卡nic` -> `tcpdump` -> `iptables(netfilter)` -> `app` -> `iptables(netfilter)` -> `tcpdump` -> `网卡nic`

显而易见，数据包到达网卡后，tcpdump有能力直接捕获到，不受iptables的影响，此时数据包还没有到达iptables的 `PREROUTING` 链，到达APP后，处理完报文从iptables出去，出去最终要走到 `POSTROUTING` 链再到tcpdump，所以此时受到iptables的 `OUTPUT` 和 `POSTROUTING` 链影响，这两条链的规则决定tcpdump能不能抓到出去的包，iptables各个链工作顺序如下(图片来源于网络)：



本文将汇总日常工作使用中的一些用法和技巧，同时附上tcpdump官方文档： **https://www.tcpdump.org/tcpdump_man.html** 。

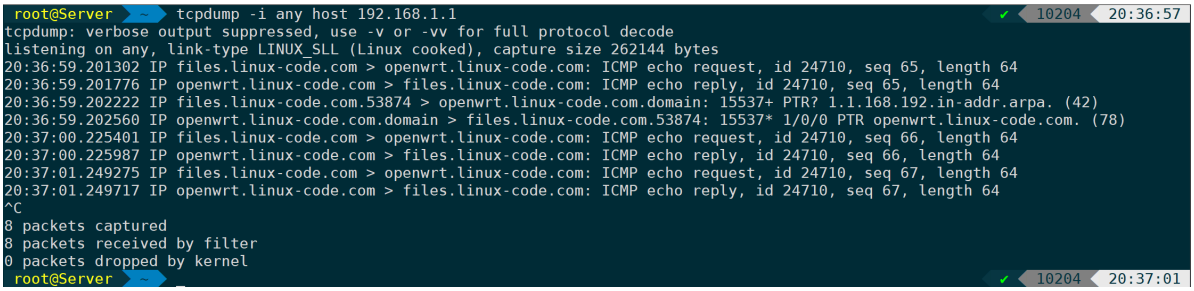
二、tcpdump常用参数详解

1.指定网卡和主机抓包(-i、host)

指定所有网卡并限定主机为192.168.1.1

```
tcpdump -i any host 192.168.1.1       #-i指定网卡为所有
```

此时如果和对端主机有数据交互，那么屏幕上有输出信息，同时这些信息不会保存在文件里，比如下面这个icmp包：



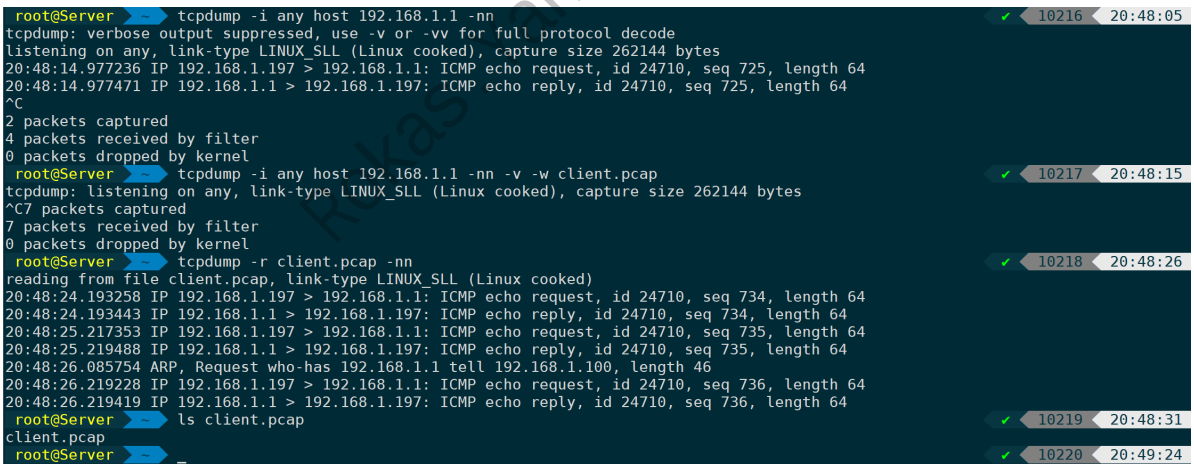
`CTRL + C` 给进程发送 `SIGINT` 信号，中断tcpdump当前抓包，会发现这些数据包默认会显示在屏幕上，如果是简单数据包直接拿tcpdump分析是没问题的，但在报文交互很大的场景下，要过滤特定流，此时用tcpdump分析效率会很低，正确的做法是tcpdump抓包保存为抓包文件(`.pcap` 、 `.cap` 都行)，再用wireshark分析。

同时这里还有问题，tcpdump默认会将IP反向解析成host或域名， `-nn` 参数可以禁止反向解析，增加可读性。

2.写入文件并查阅报文(-w、-r)

那么改进后的抓包命令可以是：

```
tcpdump -i any host 192.168.1.1 -nn -v -w client.pcap
```



`-nn` ： 禁止反向解析

`-v` ： 显示详细抓包信息

`-w` ： 写入到client.pcap

可以看到第一条命令，IP不再反解成host或域名，第二条命令 `-w` 写入到client.pcap后，使用 `-r` 参数指定报文文件查阅内容。

3.指定来源IP或目的IP或网段(src、dst、net)

如果只想抓一个方向的流，使用 `src` 指定源或者 `dst` 指定目标：

```
tcpdump -i any src host 192.168.1.1 -nn
```

```
root@Server ~ tcpdump -i any src host 192.168.1.1 -nn                    ✔ 10221 20:55:09
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
20:55:12.705469 IP 192.168.1.1 > 192.168.1.197: ICMP echo reply, id 24710, seq 1133, length 64
20:55:13.729436 IP 192.168.1.1 > 192.168.1.197: ICMP echo reply, id 24710, seq 1134, length 64
20:55:14.746354 ARP, Request who-has 192.168.1.197 tell 192.168.1.1, length 46
20:55:14.753334 IP 192.168.1.1 > 192.168.1.197: ICMP echo reply, id 24710, seq 1135, length 64
20:55:15.777424 IP 192.168.1.1 > 192.168.1.197: ICMP echo reply, id 24710, seq 1136, length 64
^C
5 packets captured
5 packets received by filter
0 packets dropped by kernel
root@Server ~ _                                                          ✔ 10222 20:55:16
```

这样一来只抓一个方向的流，如图只有1.1方向过来 `icmp reply` ，以及arp报文，arp缓存表过期后则会发送一次广播，以获得mac地址。

```
root@OpenWrt:~# ifconfig br-lan
br-lan    Link encap:Ethernet  HWaddr 00:0C:29:BE:5A:26
          inet addr:192.168.1.1  Bcast:192.168.1.0  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:febe:5a26/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:16074160 errors:0 dropped:487 overruns:0 frame:0
          TX packets:14212856 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5527697754 (5.1 GiB)  TX bytes:5483762517 (5.1 GiB)

root@OpenWrt:~# arp -a|grep 197
192.168.1.197    0x1         0x2         00:0c:29:50:c6:dc     *        br-lan
root@OpenWrt:~# _
```

`dst` 指定到达目的方向的包，过滤本端发给对端的 `icmp request` ：

```
tcpdump -nn -i any dst 192.168.1.1
```

```
root@Server ~ tcpdump -nn -i any dst 192.168.1.1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
21:01:54.049251 IP 192.168.1.197 > 192.168.1.1: ICMP echo request, id 24710, seq 1525, length 64
21:01:55.073265 IP 192.168.1.197 > 192.168.1.1: ICMP echo request, id 24710, seq 1526, length 64
21:01:56.090756 ARP, Request who-has 192.168.1.1 tell 192.168.1.100, length 46
21:01:56.097249 IP 192.168.1.197 > 192.168.1.1: ICMP echo request, id 24710, seq 1527, length 64
21:01:57.121257 IP 192.168.1.197 > 192.168.1.1: ICMP echo request, id 24710, seq 1528, length 64
^C
5 packets captured
5 packets received by filter
0 packets dropped by kernel
root@Server ~
```

指定网段：

```
tcpdump -nn -i any net 192.168.1.1/32
```

```
root@Server ~ tcpdump -nn -i any net 192.168.1.1/32
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
21:05:15.745270 IP 192.168.1.197 > 192.168.1.1: ICMP echo request, id 24710, seq 1722, length 64
21:05:15.745408 IP 192.168.1.1 > 192.168.1.197: ICMP echo reply, id 24710, seq 1722, length 64
21:05:16.769247 IP 192.168.1.197 > 192.168.1.1: ICMP echo request, id 24710, seq 1723, length 64
21:05:16.769700 IP 192.168.1.1 > 192.168.1.197: ICMP echo reply, id 24710, seq 1723, length 64
21:05:17.793322 IP 192.168.1.197 > 192.168.1.1: ICMP echo request, id 24710, seq 1724, length 64
21:05:17.793743 IP 192.168.1.1 > 192.168.1.197: ICMP echo reply, id 24710, seq 1724, length 64
^C
6 packets captured
6 packets received by filter
0 packets dropped by kernel
root@Server ~ _
```

4.指定每个报文抓包字节数(-s)

有时候我们分析包一般分析头部就行了，不需要分析每个包的data数据部分，这样可以尽量减小抓包文件的大小，`-s` 指定每个报文抓前面多少个字节：

```
tcpdump -nn -i any -s 84 host 192.168.1.1  #imcp协议默认为"56字节"数据字节+"28字节"的ICMP头，一共是84字节
```

```
root@Server ~ tcpdump -nn -i any -s 84 host 192.168.1.1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 84 bytes
21:17:42.145264 IP 192.168.1.197 > 192.168.1.1: ICMP echo request, id 24710, seq 2451, length 64
21:17:42.145468 IP 192.168.1.1 > 192.168.1.197: ICMP echo reply, id 24710, seq 2451, length 64
21:17:43.169270 IP 192.168.1.197 > 192.168.1.1: ICMP echo request, id 24710, seq 2452, length 64
21:17:43.169434 IP 192.168.1.1 > 192.168.1.197: ICMP echo reply, id 24710, seq 2452, length 64
^C
4 packets captured
4 packets received by filter
0 packets dropped by kernel
root@Server ~
```

如果不指定 `-s` 参数，tcpdump默认只会抓每个报文的前面56个字节，`-s 0` 可以不限制字节数，每次都抓完整的包：

```
tcpdump -nn -i any -s 0 host 192.168.1.1
```

如果是icmp，56+8个字节的icmp头部，默认是64个字节：

```
root@Server ~ man ping|grep -w -A 2 -- '^\s*-s'
       -s packetsize
              Specifies the number of data bytes to be sent.  The default is 56, which translates into 64 ICMP data bytes when combined with the 8 bytes of ICMP header data.

root@Server ~ _
```

5.指定端口和协议(port、portrange、[protocol])

全量抓不如抓出问题的端口或协议，对症下药，语法也很简单：

```
tcpdump -nn -i any -s 0 icmp  #只抓icmp协议
```

```
root@Server ~ tcpdump -nn -i any -s 0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
21:27:48.289260 IP 192.168.1.197 > 192.168.1.1: ICMP echo request, id 24710, seq 3043, length 64
21:27:48.289406 IP 192.168.1.1 > 192.168.1.197: ICMP echo reply, id 24710, seq 3043, length 64
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel
root@Server ~ _
```

icmp运行在网络层偏上，所以并没有端口的定义，抓icmp并不需要指定端口，那么抓tcp端口可以这样写：

```
tcpdump -nn -i any -s 60 tcp port 80  #作为演示，这里只抓60个头部字节
```

```
root@Server ~ tcpdump -nn -i any -s 60 tcp port 80                              ✔ 10249 21:32:37
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 60 bytes
21:35:37.990852 IP 192.168.1.197.48244 > 192.168.1.1.80: Flags [S], seq 1003356661, win 29200, options [mss 1460,[|tcp]>
21:35:37.991220 IP 192.168.1.1.80 > 192.168.1.197.48244: Flags [S.], seq 323433920, ack 1003356662, win 65535, options [mss 1460,[|
tcp]>
21:35:37.991261 IP 192.168.1.197.48244 > 192.168.1.1.80: Flags [.], ack 1, win 115, options [nop,nop,TS[|tcp]>
21:35:38.728389 IP 192.168.1.197.48244 > 192.168.1.1.80: Flags [P.], seq 1:6, ack 1, win 115, options [nop,nop,TS[|tcp]>
21:35:38.728848 IP 192.168.1.1.80 > 192.168.1.197.48244: Flags [.], ack 6, win 512, options [nop,nop,TS[|tcp]>
21:35:39.740351 IP 192.168.1.1.80 > 192.168.1.197.48244: Flags [.], ack 6, win 512, options [nop,nop,TS[|tcp]>
21:35:39.740395 IP 192.168.1.197.48244 > 192.168.1.1.80: Flags [.], ack 1, win 115, options [nop,nop,TS[|tcp]>
^C
7 packets captured
7 packets received by filter
0 packets dropped by kernel
root@Server ~ _                                                                  ✔ 10251 21:35:40
```

同理，那么udp可以这样写：

```
tcpdump -nn -i any -s 0 udp port 22
```

```
 root@Server  ~  tcpdump -nn -i any -s 0 udp port 22
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
21:37:28.156710 IP 192.168.1.197.42189 > 192.168.1.1.22: UDP, length 1
21:37:32.268648 IP 192.168.1.197.42662 > 192.168.1.1.22: UDP, length 1
21:37:32.724513 IP 192.168.1.197.57118 > 192.168.1.1.22: UDP, length 1
^C
3 packets captured
4 packets received by filter
0 packets dropped by kernel
 root@Server  ~
```

抓端口范围，需要用到 `portrange` 参数：

```
tcpdump -nn -i any tcp portrange 53-80
```

```
 root@Server  ~  tcpdump -nn -i any tcp portrange 53-80                                                    ✔  10466  01:43:25
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
01:43:33.821494 IP 192.168.1.197.63036 > 192.168.1.1.53: Flags [S], seq 3040685040, win 1024, options [mss 1460], length 0
01:43:33.821528 IP 192.168.1.197.63036 > 192.168.1.1.80: Flags [S], seq 3040685040, win 1024, options [mss 1460], length 0
01:43:33.821848 IP 192.168.1.1.80 > 192.168.1.197.63036: Flags [S.], seq 2450991194, ack 3040685041, win 65535, options [mss 1460]
, length 0
01:43:33.821884 IP 192.168.1.197.63036 > 192.168.1.1.80: Flags [R], seq 3040685041, win 0, length 0
01:43:33.821896 IP 192.168.1.1.53 > 192.168.1.197.63036: Flags [S.], seq 739381406, ack 3040685041, win 65535, options [mss 1460],
 length 0
01:43:33.821906 IP 192.168.1.197.63036 > 192.168.1.1.53: Flags [R], seq 3040685041, win 0, length 0
^C
6 packets captured
6 packets received by filter
0 packets dropped by kernel
 root@Server  ~                                                                                           ✔  10467  01:43:35
```

如果不确定协议，只指定 `port` 即可：

```
tcpdump -nn -i any -s 0 port 22
```

6.tcpdump的逻辑表达式(or、and、not)

逻辑语句，顾名思义，只要接触过一点编程，就知道指的是或与非，及 `or` 、 `and` 、 `not` ，其中 `not` 也可以用作 `!` ，此三个参数较常用，可以帮我们过滤出有用的信息，指哪打哪。

使用 `and` 指定目标和协议：

```
tcpdump -nn -i any -s 0 host 192.168.1.1 and icmp
```

```
 root@Server  ~  tcpdump -nn -i any -s 0 host 192.168.1.1 and icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
21:44:55.297260 IP 192.168.1.197 > 192.168.1.1: ICMP echo request, id 29760, seq 31, length 64
21:44:55.297828 IP 192.168.1.1 > 192.168.1.197: ICMP echo reply, id 29760, seq 31, length 64
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel
 root@Server  ~
```

使用 `or` 指定多个过滤条件：

```
tcpdump -nn -i any -s 0 host 192.168.1.1 or icmp or src net 192.168.1.1/32
```

```
 root@Server  ~  tcpdump -nn -i any -s 0 host 192.168.1.1 or icmp or src net 192.168.1.1/32
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
21:47:55.393244 IP 192.168.1.197 > 192.168.1.1: ICMP echo request, id 29760, seq 207, length 64
21:47:55.393384 IP 192.168.1.1 > 192.168.1.197: ICMP echo reply, id 29760, seq 207, length 64
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel
 root@Server  ~
```

使用 `or` 或 `!` 排除过滤条件：

```
tcpdump -nn -i any -s 0 ! net 172.16.0.0/16 and icmp and ! tcp
```

```
root@Server ~ tcpdump -nn -i any -s 0 ! net 172.16.0.0/16 and icmp and ! tcp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
12:25:56.673270 IP 192.168.1.197 > 192.168.1.1: ICMP echo request, id 6398, seq 65, length 64
12:25:56.673473 IP 192.168.1.1 > 192.168.1.197: ICMP echo reply, id 6398, seq 65, length 64
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel
root@Server ~ _
```

指定要抓包的网段掩码即可，那么综合以上，你可以举一反三，把它们搭配起来灵活运用，只要抓包逻辑语法正确即可，哪怕又臭又长：

```
tcpdump -nn -i any -s 0 dst host 192.168.1.197 and icmp and src net 192.168.1.1/32 or \( host 192.168.1.1 \) and ! tcp
```

```
root@Server ~ tcpdump -nn -i any -s 0 dst host 192.168.1.197 and icmp and src net 192.168.1.1/32 or \( host 192.168.1.1 \) and ! tcp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
12:26:52.993271 IP 192.168.1.197 > 192.168.1.1: ICMP echo request, id 6398, seq 120, length 64
12:26:52.993462 IP 192.168.1.1 > 192.168.1.197: ICMP echo reply, id 6398, seq 120, length 64
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel
root@Server ~ _
```

bash解析每条命令最多255个字符，又臭又长无意义的语句不是拿来用于生产环境，而是平时练手可以多拼凑组合测试，到达炉火纯青的那一刻，生产环境想抓什么包想怎么搭配使用信手拈来。

7.指定数据包大小过滤(greater、less)

tcpdump提供指定每个报文大小的过滤方式，在需要过滤大包或特定大小的包场景特别好用。

过滤大于1000字节的包：

```
tcpdump -nn -s 0 -i any host 192.168.1.1 and greater 1000 and icmp
```

```
root@Server ~ tcpdump -nn -s 0 -i any host 192.168.1.1 and greater 1000 and icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
12:36:15.713241 IP 192.168.1.197 > 192.168.1.1: ICMP echo request, id 8544, seq 8, length 1032
12:36:15.713576 IP 192.168.1.1 > 192.168.1.197: ICMP echo reply, id 8544, seq 8, length 1032
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel
root@Server ~ _
```

过滤小于1000字节的包：

```
tcpdump -nn -s 0 -i any host 192.168.1.1 and less 1000 and icmp
```

```
root@Server ~ tcpdump -nn -s 0 -i any host 192.168.1.1 and less 1000 and icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
12:41:43.425257 IP 192.168.1.197 > 192.168.1.1: ICMP echo request, id 8968, seq 8, length 708
12:41:43.425728 IP 192.168.1.1 > 192.168.1.197: ICMP echo reply, id 8968, seq 8, length 708
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel
root@Server ~ _
```

结合使用，缩小范围：

```
tcpdump -nn -s 0 -i any host 192.168.1.1 and less 800 and greater 690
```

```
 root@Server  ~  tcpdump -nn -s 0 -i any host 192.168.1.1 and less 800 and greater 690
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
12:44:59.425244 IP 192.168.1.197 > 192.168.1.1: ICMP echo request, id 9725, seq 59, length 700
12:44:59.425459 IP 192.168.1.1 > 192.168.1.197: ICMP echo reply, id 9725, seq 59, length 700
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel
 root@Server  ~  _
```

8.Flags标记解读

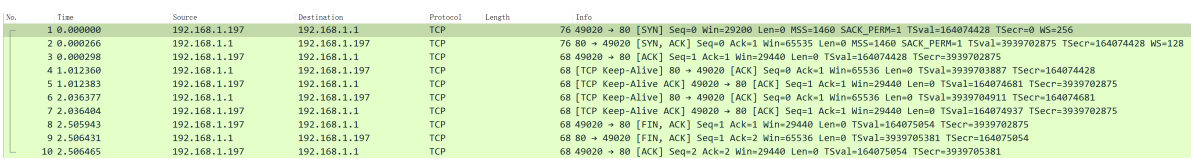
下面的命令抓取一个完整连接，包括三次握手和四次挥手：

```
 root@Server  ~  tcpdump -i any -nn -s 0 host 192.168.1.1 and tcp port 80                                    ✔  10396  12:56:37
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
12:56:46.062835 IP 192.168.1.197.49008 > 192.168.1.1.80: Flags [S], seq 2205954638, win 29200, options [mss 1460,sackOK,TS val 163
919883 ecr 0,nop,wscale 8], length 0
12:56:46.063175 IP 192.168.1.1.80 > 192.168.1.197.49008: Flags [S.], seq 24614250, ack 2205954639, win 65535, options [mss 1460,sa
ckOK,TS val 3939084697 ecr 163919883,nop,wscale 7], length 0
12:56:46.063210 IP 192.168.1.197.49008 > 192.168.1.1.80: Flags [.], ack 1, win 115, options [nop,nop,TS val 163919883 ecr 39390846
97], length 0
12:56:47.076868 IP 192.168.1.1.80 > 192.168.1.197.49008: Flags [.], ack 1, win 512, options [nop,nop,TS val 3939085711 ecr 1639198
83], length 0
12:56:47.076905 IP 192.168.1.197.49008 > 192.168.1.1.80: Flags [.], ack 1, win 115, options [nop,nop,TS val 163920136 ecr 39390846
97], length 0
12:56:47.335323 IP 192.168.1.197.49008 > 192.168.1.1.80: Flags [F.], seq 1, ack 1, win 115, options [nop,nop,TS val 163920201 ecr
3939084697], length 0
12:56:47.337030 IP 192.168.1.1.80 > 192.168.1.197.49008: Flags [F.], seq 1, ack 2, win 512, options [nop,nop,TS val 3939085971 ecr
 163920201], length 0
12:56:47.337068 IP 192.168.1.197.49008 > 192.168.1.1.80: Flags [.], ack 2, win 115, options [nop,nop,TS val 163920201 ecr 39390859
71], length 0
^C
8 packets captured
8 packets received by filter
0 packets dropped by kernel
 root@Server  ~  _                                                                                           ✔  10396  12:56:50
```

每个Flags含义如下：

Flags	含义
`[S]`	SYN
`[.]`	ACK
`[S.]`	SYN、ACK
`[P.]`	PUSH
`[R.]`	RST
`[F.]`	FIN
`[DF]`	Don't Fragment(不分片)，当DF=0时，允许分片
`[FP.]`	FIN、PUSH、ACK

前面说过，如果使用tcpdump不好直观分析报文，可`-w`保存到文件后使用wireshark分析：



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.197	192.168.1.1	TCP	76	49020 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=164074428 TSecr=0 WS=256
2	0.000266	192.168.1.1	192.168.1.197	TCP	76	80 → 49020 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=3939702875 TSecr=164074428 WS=128
3	0.000298	192.168.1.197	192.168.1.1	TCP	68	49020 → 80 [ACK] Seq=1 Ack=1 Win=29440 Len=0 TSval=164074428 TSecr=3939702875
4	1.012360	192.168.1.1	192.168.1.197	TCP	68	[TCP Keep-Alive] 80 → 49020 [ACK] Seq=0 Ack=1 Win=65536 Len=0 TSval=3939703887 TSecr=164074428
5	1.012383	192.168.1.197	192.168.1.1	TCP	68	[TCP Keep-Alive ACK] 49020 → 80 [ACK] Seq=1 Ack=1 Win=29440 Len=0 TSval=164074681 TSecr=3939702875
6	2.036377	192.168.1.1	192.168.1.197	TCP	68	[TCP Keep-Alive] 80 → 49020 [ACK] Seq=0 Ack=1 Win=65536 Len=0 TSval=3939704911 TSecr=164074681
7	2.036404	192.168.1.197	192.168.1.1	TCP	68	[TCP Keep-Alive ACK] 49020 → 80 [ACK] Seq=1 Ack=1 Win=29440 Len=0 TSval=164074937 TSecr=3939702875
8	2.505943	192.168.1.197	192.168.1.1	TCP	68	49020 → 80 [FIN, ACK] Seq=1 Ack=1 Win=29440 Len=0 TSval=164075054 TSecr=3939702875
9	2.506431	192.168.1.1	192.168.1.197	TCP	68	80 → 49020 [FIN, ACK] Seq=1 Ack=2 Win=65536 Len=0 TSval=3939705381 TSecr=164075054
10	2.506465	192.168.1.197	192.168.1.1	TCP	68	49020 → 80 [ACK] Seq=2 Ack=2 Win=29440 Len=0 TSval=164075054 TSecr=3939705381

建立三次握手后，`Tcp Keep-Alive`一直保留连接，之后客户端主动挥手断开连接，客户端对应的行为是这样的：

```
root@Server ~ telnet 192.168.1.1 80
Trying 192.168.1.1...
Connected to 192.168.1.1.
Escape character is '^]'.
^]
telnet> quit
Connection closed.
root@Server ~ _
```

9.指定抓包数量、抓包大小、及轮询抓包(-c、-W、-C、-G)

在某些场景需要分割抓包文件，轮询抓包，那么这几个参数可以排上用上：

- `-c` 指定抓多少个包
- `-W` 最多写入多少个抓包文件，以MB为单位
- `-C` 写入到抓包文件的大小上限

```
root@Server ~ man tcpdump |& grep -P -w -A 2 '^\s*(-c|-W|-C|-G)'                ✔ 10504 02:08:34
       -c count
              Exit after receiving count packets.

       -C file_size
              Before writing a raw packet to a savefile, check whether the file is currently larger than file_size and, if so,
              close  the current savefile and open a new one.  Savefiles after the first savefile will have the name specified
--
       -G rotate_seconds
              If specified, rotates the dump file specified with the -w option every rotate_seconds seconds.   Savefiles  will
              have  the  name specified by -w which should include a time format as defined by strftime(3).  If no time format
--
       -W     Used in conjunction with the -C option, this will limit the number of files created to the specified number, and
              begin  overwriting  files  from the beginning, thus creating a 'rotating' buffer.  In addition, it will name the
              files with enough leading 0s to support the maximum number of files, allowing them to sort correctly.
root@Server ~ _                                                                  ✔ 10505 02:08:38
```

`-c` 指定抓2个包：

```
tcpdump -i any -s 0 net 192.168.1.1/32 -c 2
```

```
root@Server ~ tcpdump -i any -s 0 net 192.168.1.1/32 -c 2                        ✔ 10423 01:21:17
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
01:22:37.889257 IP files.linux-code.com > openwrt.linux-code.com: ICMP echo request, id 21746, seq 33, length 64
01:22:37.889427 IP openwrt.linux-code.com > files.linux-code.com: ICMP echo reply, id 21746, seq 33, length 64
2 packets captured
4 packets received by filter
0 packets dropped by kernel
root@Server ~ _                                                                  ✔ 10425 01:22:37
```

`-C` 指定写入到文件的大小上限为1M：

```
tcpdump -i any -s 0 -C 1 -v -w client.pcap
```

`-W` 指定写入到10个抓包文件，每个文件只抓1M，循环写入：

```
tcpdump -i any -s 0 -C 1M -v -W 10 -w client.pcap
```

```
 root@Server  ~/tcpdump  tcpdump -i any -s 0 -C 1M -W 10 -v -w client.pcap
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
^C4580039 packets captured
4581595 packets received by filter
0 packets dropped by kernel
 root@Server  ~/tcpdump  ls -lh
total 9.5M
-rw-r--r-- 1 root root 977K Aug 10 01:35 client.pcap0
-rw-r--r-- 1 root root 838K Aug 10 01:35 client.pcap1
-rw-r--r-- 1 root root 977K Aug 10 01:35 client.pcap2
-rw-r--r-- 1 root root 977K Aug 10 01:35 client.pcap3
-rw-r--r-- 1 root root 977K Aug 10 01:35 client.pcap4
-rw-r--r-- 1 root root 977K Aug 10 01:35 client.pcap5
-rw-r--r-- 1 root root 977K Aug 10 01:35 client.pcap6
-rw-r--r-- 1 root root 977K Aug 10 01:35 client.pcap7
-rw-r--r-- 1 root root 977K Aug 10 01:35 client.pcap8
-rw-r--r-- 1 root root 977K Aug 10 01:35 client.pcap9
 root@Server  ~/tcpdump  _
```

保存格式将在文件后缀加入从0-N的编号。

`-G` 参数指定间隔多少秒轮询保存一次文件，通常是以时间格式命令：

```
tcpdump -nn -i any -s 0 -G 5 -Z root -v -w %m-%d-%H:%M:%S.pcap  #每隔五秒保存一次文件
```

```
 root@Server  ~/tcpdump  tcpdump -nn -i any -s 0 -G 5 -Z root -v -w %m-%d-%H:%M:%S.pcap
dropped privs to root
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
^C115 packets captured
118 packets received by filter
0 packets dropped by kernel
 root@Server  ~/tcpdump  ls
08-10-01:53:22.pcap  08-10-01:53:27.pcap  08-10-01:53:32.pcap
 root@Server  ~/tcpdump  _
```

`-Z` 参数指定每次写入新文件用 `root` 权限执行， `-w` 的时间格式为 `date` 命令的时间格式取值，可以之间拿来用。

`-G` 通常配合 `-C` 来使用，指定每次每个文件抓固定大小：

```
tcpdump -nn -i any -s 0 -C 1 -G 5 -v -w %m-%d-%H:%M:%S.pcap
```

同时也可以配合 `timeout` 命令使用，抓固定时间后停止抓包：

```
timeout 10 tcpdump -nn -i any -s 0 -G 5 -v -w %m-%d-%H:%M:%S.pcap
```

```
 root@Server  ~/tcpdump  timeout 10 tcpdump -nn -i any -s 0 -G 5 -v -w %m-%d-%H:%M:%S.pcap
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
80 packets captured
88 packets received by filter
0 packets dropped by kernel
 root@Server  ~/tcpdump  ls
08-10-02:02:58.pcap  08-10-02:03:03.pcap
 root@Server  ~/tcpdump  _
```

只抓10s，每间隔5s轮询保存一次文件。

10.抓取指定Flag位的报文

抓指定标志位虽然不常用，通常是使用wireshark分析过滤，但如果这个功能在tcpdump上使用熟练，知道自己想要什么包，那么tcpdump这一层就能筛选出对你有用的报文，减少不必要的报文量。

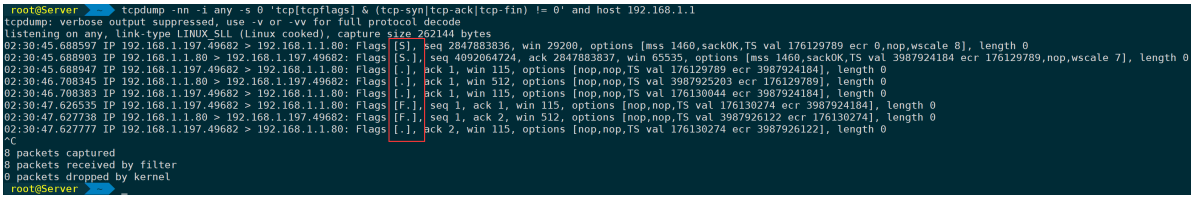
只抓SYN标志位不为0的，并且只抓一个包：

```
tcpdump -nn -i any -s 0 -c 1 'tcp[tcpflags] & tcp-syn != 0'
```

```
 root@Server  ~  tcpdump -nn -i any -s 0 -c 1 'tcp[tcpflags] & tcp-syn != 0'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
02:27:03.289900 IP 192.168.1.1.80 > 192.168.1.197.39358: Flags [S.], seq 2911228764, ack 1608983876, win 65535, options [mss 1460], length 0
1 packet captured
2 packets received by filter
0 packets dropped by kernel
 root@Server  ~
```

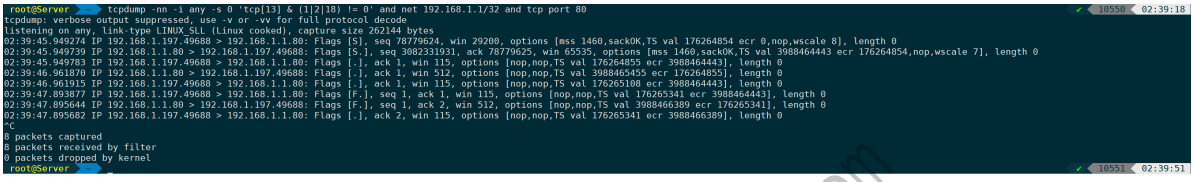
抓一次完整握手需要的标志位：

```
tcpdump -nn -i any -s 0 'tcp[tcpflags] & (tcp-syn|tcp-ack|tcp-fin) != 0' and host 192.168.1.1
```



同时，也可以用二进制表示，不过可读性较差：

```
tcpdump -nn -i any -s 0 'tcp[13] & (1|2|18) != 0' and net 192.168.1.1/32 and tcp port 80
```

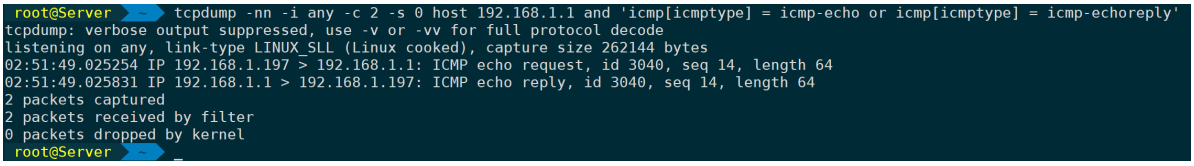


tcp字段在tcp头第十四字节存储，编程里面都是从0开始计数，那么tcp[13]就是表示tcp头部字段，附上过滤常用的几个字段含义：

表达式	含义
`tcp[13] = 2`	捕获SYN包，二进制为：00000010
`tcp[23] = 18`	捕获回给SYN的ACK包，二进制为：00010010
`tcp[13] & 2 = 2`	捕获SYN和SYN对应的ACK包
`tcp[13] = 24`	捕获PSH-ACK包
`tcp[13] & 1 = 1`	捕获FIN-ACK包
`tcp[13] & 4 = 4`	捕获RST包

其他协议也是以此类推，比如icmp：

```
tcpdump -nn -i any -c 2 -s 0 host 192.168.1.1 and 'icmp[icmptype] = icmp-echo or icmp[icmptype] = icmp-echoreply'
```



三、总结

以上总结的是一些工作中较常用的使用方法及技巧，在不同的问题场景下，知道抓什么包，明确自己需要什么包，包抓的越精准，定位问题就不会像大海捞针一样，分析包的过程也会变的会事半功倍、气定神闲。