# 概述

tcpdump 是一款 Linux 平台的抓包工具。它可以抓取涵盖整个 TCP/IP 协议族的数据包，支持针对网络层、协议、主机、端口的过滤，并提供 and、or、not 等逻辑语句来过滤无用的信息。它可以打印所有经过网络接口的数据包的头信息，也可以使用-W选项将数据包保存到文件中，方便后续分析。

# 命令

## 选项

tcpdump 有很多命令选项，想了解所有选项可以 Linux 命令行输入 tcpdump-h， man tcpdump 查看每个选项的意思。

1. [root@by ~]# tcpdump -h
2. tcpdump version 4.9.2
3. libpcap version 1.5.3
4. OpenSSL 1.0.2k-fips 26 Jan 2017
5. Usage: tcpdump [-aAbdDefhHIJKlLnNOpqStuUvxX#] [ -B size ] [ -c count ]
6. [ -C file\_size ] [ -E algo:secret ] [ -F file ] [ -G seconds ]
7. [ -i interface ] [ -j tstamptype ] [ -M secret ] [ --number ]
8. [ -Q|-P in|out|inout ]
9. [ -r file ] [ -s snaplen ] [ --time-stamp-precision precision ]
10. [ --immediate-mode ] [ -T type ] [ --version ] [ -V file ]
11. [ -w file ] [ -W filecount ] [ -y datalinktype ] [ -z postrotate-command ]
12. [ -Z user ] [ expression ]

下面列举一些常用选项：

-A 只使用ASCII打印报文的全部数据，不要和-X一起使用，获取http可以用 tcpdump-nSA port80

-b 在数据链路层上选择协议，包括 ip, arp, rarp, ipx 等

-c 指定要抓取包的数量

-D 列出操作系统所有可以用于抓包的接口

-i 指定监听的网卡， -i any显示所有网卡

-n 表示不解析主机名，直接用IP显示，默认是用hostname显示

-nn 表示不解析主机名和端口，直接用端口号显示，默认显示是端口号对应的服务名

-p 关闭接口的混杂模式

-P 指定抓取的包是流入的包还是流出的，可以指定参数 in, out, inout 等，默认是inout

-q 快速打印输出，即只输出少量的协议相关信息

-s len 设置要抓取数据包长度为len，默认只会截取前96bytes的内容，-s0的话，会截取全部内容。

-S 将TCP的序列号以绝对值形式输出，而不是相对值

-t 不要打印时间戳

-vv 输出详细信息（比如 tos、ttl、checksum等）

-X 同时用hex和ascii显示报文内容

-XX 同-X，但同时显示以太网头部

注：

常用的两个命令：

1. 抓取主机特定端口的索引ASCII报文

tcpdump -i any host \*\*\*\* and port \*\*\*\* -A > \*\*\*\*.log

注：这种方式抓取的可以直接用vim打开，如果需要用wireshark打开，则需要使用：

1. 抓取wireshark打开的报文

tcpdump -i eth0 -s 0 -w \*\*\*\*.cap -W 500 -C 100 tcp port \*\*\*\* and host \*\*\*\*

## 过滤器

网络报文是很多的，很多时候我们在主机上抓包，会抓到很多我们并不关心的无用包，然后要从这些包里面去找我们需要的信息，无疑是一件费时费力的事情，tcpdump提供了灵活的语法可以精确获取我们关心的数据，这些语法说得专业点就是过滤器。

过滤器简单可分为三类：协议（proto）、传输方向（dir）和类型（type）。

一般的表达式格式为：

关于proto：可选有ip、arp、rarp、tcp、udp、icmp、ether等，默认是所有协议的包

关于dir：可选有src、dst、src or dst、src and dst，默认为src or dst

关于type：可选有 host、net、port、portrange（端口范围，比如21-42），默认为host

# 常用操作

测试环境 IP：172.18.82.173

## 抓取某主机的数据包

抓取主机 172.18.82.173 上所有收到（DST\_IP）和发出（SRC\_IP）的所有数据包

tcpdump host 172.18.82.173

抓取经过指定网口 interface ，并且 DST\_IP 或 SRC\_IP 是 172.18.82.173 的数据包

tcpdump -i eth0 host 172.18.82.173

筛选 SRC\_IP，抓取经过 interface 且从 172.18.82.173 发出的包

tcpdump -i eth0 src host 172.18.82.173

筛选 DST\_IP，抓取经过 interface 且发送到 172.18.82.173 的包

tcpdump -i eth0 dst host 172.18.82.173

抓取主机 200.200.200.1 和主机 200.200.200.2 或 200.200.200.3 通信的包

tcpdump host 200.200.200.1 and \(200.200.200.2 or 200.200.200.3\)

抓取主机 200.200.200.1 和除了主机 200.200.200.2 之外所有主机通信的包

tcpdump ip host 200.200.200.1 and ! 200.200.200.2

## 抓取某端口的数据包

抓取所有端口，显示 IP 地址

tcpdump -nS

抓取某端口上的包

tcpdump port 22

抓取经过指定 interface，并且 DST\_PORT 或 SRC\_PORT 是 22 的数据包

tcpdump -i eth0 port 22

筛选 SRC\_PORT

tcpdump -i eth0 src port 22

筛选 DST\_PORT

tcpdump -i eth0 dst port 22

比如希望查看发送到 host 172.18.82.173 的网口eth0的22号端口的包

[root@by ~]# tcpdump -i eth0 -nnt dst host 172.18.82.173 and port 22 -c 1 -vv

tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes

IP (tos 0x14, ttl 114, id 27674, offset 0, flags [DF], proto TCP (6), length 40)

113.98.59.61.51830 > 172.18.82.173.22: Flags [.], cksum 0x7fe3 (correct), seq 19775964, ack 1564316089, win 2053, length 0

## 抓取某网络（网段）的数据包

抓取经过指定 interface，并且 DST\_NET 或 SRC\_NET 是 172.18.82 的包

tcpdump -i eth0 net 172.18.82

筛选 SRC\_NET

tcpdump -i eth0 src net 172.18.82

筛选 DST\_NET

tcpdump -i eth0 dst net 172.18.82

## 抓取某协议的数据包

tcpdump -i eth0 icmp

tcpdump -i eth0 ip

tcpdump -i eth0 tcp

tcpdump -i eth0 udp

tcpdump -i eth0 arp

## 复杂的逻辑表达式抓取过滤条件

抓取经过 interface eth0 发送到 host 200.200.200.1 或 200.200.200.2 的 TCP 协议 22 号端口的数据包

tcpdump -i eth0 -nntvv -c 10 '((tcp) and (port 22) and ((dst host 200.200.200.1) or (dst host 200.200.200.2)))'

注：对于复杂的过滤器表达式，为了逻辑清晰，可以使用 ()，不过默认情况下，tcpdump 会将()当做特殊字符，所以必须使用 '' 来消除歧义。

抓取经过 interface eth0， DST\_MAC 或 SRC\_MAC 地址是 00:16:3e:12:16:e7 的 ICMP 数据包

tcpdump -i eth0 '((icmp) and ((ether host 00:16:3e:12:16:e7)))' -nnvv

抓取经过 interface eth0，目标网络是 172.18 但目标主机又不是 172.18.82.173 的 TCP 且非 22 号端口号的数据包

tcpdump -i eth0 -nntvv '((dst net 172.18) and (not dst host 172.18.82.173) and (tcp) and (not port 22))'

抓取流入 interface eth0，host 为 172.18.82.173 且协议为 ICMP 的数据包

tcpdump -i eth0 -nntvv -P in host 172.18.82.173 and icmp

抓取流出 interface eth0，host 为 172.18.82.173 且协议为 ICMP 的数据包

tcpdump -i eth0 -nntvv -P out host 172.18.82.173 and icmp

# Wireshark

tcpdump抓包的时候，默认是打印到屏幕输出，如果是抓取包少还好，如果包很多，很多行数据，刷刷刷从眼前一闪而过，根本来不及看清内容。不过，tcpdump提供了将抓取的数据保存到文件的功能，查看文件就方便分析多了，而且还能与其他图形工具一起配合分析，比如wireshark、Snort等。

-w 选项表示把数据报文输出到文件

tcpdump -w capture\_file.pcap port 80 # 把所有 80 端口的数据导出到文件

-r 选项表示读取文件里的数据报文，显示到屏幕上

tcpdump -nXr capture\_file.pcap host host1

注：.pcap 格式的文件需要用wireshark、Snort 等工具查看，使用vim或cat会出现乱码。

# 实战

我们可以使用tcpdump抓包，然后使用fiddle或者wireshark分析数据包。

比如我们需要抓取10.47.181.1这台主机的br\_com网桥发往10.47.181.2主机8080端口的数据包，如下操作：

（1）抓包

tcpdump –i br\_com src 10.47.181.1 and dst 10.47.181.2 and port 8080 –w test.cap

（2）分析

转包后的文件用wireshark打开，点击对应的http包，然后点击追踪HTTP流，就可以看到整个发送包的内容。

## 虚机抓包

1. virsh list
2. 11 uuid
3. virsh domiflist id(11) 这时候可以看到
4. interface type source model mac
5. vnet11.0 bridge br0 rtl8139 ----
6. 在自己的虚拟机上ifconfig查看ip地址 10.47.118.114
7. ping ip -t
8. brctl show br0主机上查看网桥br0的结构

9、tcpdump –envvi eno1 icmp | grep 10.47.148.114 在主机的网卡上捕捉从这个ip发过来的icmp包

10、tcpdump –envvi vnet11.0 icmp | grep 10.47.148.114 在虚机的网卡上捕获发来的icmp

11、brctl showmacs br0查看br0的mac

12、有一种确定虚机的网卡ip的方法(interface) 在主机ping 10.47.181.32然后通过tcpdump抓包，尝试对每个interface，看那个interface抓到包，就表明这个interface与ip相对应

## 主机抓包

Linux下抓包实践：

tcpdump –I br\_comm src ip1 and dst ip2 and port 8081 –w test.cap

转包后的文件test.cap用wireshark打开

点击对应的http包，然后点击追踪http流，就可以看到整个发送的包

tcpdump –envvi threeport dst ip and port 081

# 分布式数据库实践

在分布式数据库中，遇到这样一种情况，客户端连接计算节点proxy时报错：

You can change this value on the server by setting the max\_allowed\_packet variables;

直接连接DB数据节点可以正常执行，并且在java连接串中显式增加max\_allowed\_packet（16M）可以消除该报错。

**排查思路：**

1. 确认JDBC的版本号：5.1.46
2. 分别抓包：客户端->计算节点proxy，计算节点proxy->数据节点DB
3. JDBC连接串显式增加max\_allowed\_packet，抓包
4. 分析步骤2、3抓包中包的不同，发现存在一个问题：JDBC显式设置max\_allowed\_packet的时候传过来的是大写参数，而不显示设置参数则传递到proxy的是小写参数，经过确认，JDBC是区分大小写的，所以小写的就没有匹配成功，参数失效
5. 经过业务代码确认，这个应该是Oracle开关（parse\_mode）设置不同导致的