**tcpdump**

**[功能]**

抓包工具。

**[描述]**

**1、简介**

tcpdump，就是：dump the traffice on a network，是Linux上的网络数据采集分析工具（即sniffer）， 可以根据使用者的定义对网络上的数据包进行截获的包分析工具。sniffer工具首先是为网络管理员和网络程序员进行网络分析而设计的。对于网络管理人员来说，使用嗅探器可以随时掌握网络的实际情况，在网络性能急剧下降的时候，可以通过sniffer工具来分析原因，找出造成网络阻塞的来源。对于网络程序员来说,通过sniffer工具来调试程序。

**2、选项**

tcpdump支持相当多的参数，如使用-i参数指定tcpdump监听的网络界面，这在计算机具有多个网络界面时非常有用，使用-c参数指定要监听的数据包数量，使用-w参数指定将监听到的数据包写入文件中保存，等等。如下：

-a 将网络地址和广播地址转变成名字；

-b 在数据-链路层上选择协议，包括ip、arp、rarp、ipx都是这一层的。

-c 指定要监听的数据包数量，在收到指定的包的数目后，tcpdump就会停止.

-d 将匹配信息包的代码以人们能够理解的汇编格式给出；

-dd 将匹配信息包的代码以c语言程序段的格式给出；

-ddd 将匹配信息包的代码以十进制的形式给出；

-e 在输出行打印出数据链路层的头部信息；

-F 从指定的文件中读取表达式,忽略其它的表达式；

-f 将外部的Internet地址以数字的形式打印出来；

-i 指定监听的网络接口，这在计算机具有多个网络接口时非常有用.

-l 用于输出为行缓冲的形式，可以将数据重定向。

-N 不打印出默认的域名。前面信息-N后就是：eth0 < ntc9.1165 > router.telnet。

-n 不进行IP地址到主机名的转换。如果不使用这一项，当系统中存在某一主机的主机名时，TcpDump会把IP地址转换为主机名显示，就像这样：eth0 < ntc9.1165> router.domain.net.telnet，使用-n后变成了：eth0 < 192.168.0.9.1165 > 192.168.0.1.telnet。

-nn 不进行端口名称的转换。前面那条信息使用-nn后就变成了：eth0 < ntc9.1165 > router.domain.net.23。

-O 不进行匹配代码的优化。

-r 从指定的文件中读取包(这些包一般通过-w选项产生)；

-T 将监听到的包直接解释为指定的类型的报文，常见的类型有rpc （远程过程调用）和snmp（简单网络管理协议；）

-t 在输出的每一行不打印时间戳；

-tt 打印原始的、未格式化过的时间。

-v 输出一个稍微详细的信息，例如在ip包中可以包括ttl和服务类型的信息；

-vv 输出详细的报文信息；

-w 指定将监听到的数据包写入文件中保存而并不分析和打印出来.

**3、过滤条件**

tcpdump使用参数指定要监视数据包的类型、地址、端口等，根据具体的网络问题，更复杂的tcpdump参数是用于过滤目的，充分利用这些过滤规则就能达到迅速定位故障的目的。因为网络中流量很大，如果不加分辨将所有的数据包都截留下来，数据量太大，反而不容易发现需要的数据包，使用这些参数定义的过滤规则可以截留特定的数据包，以缩小目标，更好的分析网络中存在的问题。请使用man tcpdump查看这些过滤规则的具体用法。

显然为了安全起见，不用作网络管理用途的计算机上不应该运行这一类的网络分析软件，为了屏蔽它们，可以屏蔽内核中的bpfilter伪设备。一般情况下网络硬件和TCP/IP堆栈不支持接收或发送与本计算机无关的数据包，为了接收这些数据包，就必须使用网卡的混杂模式，并绕过标准的TCP/IP 堆栈才行。在FreeBSD下，这就需要内核支持伪设备bpfilter。因此，在内核中取消bpfilter支持，就能屏蔽tcpdump之类的网络分析工具。当网卡被设置为混杂模式时，系统会在控制台和日志文件中留下类似如下记录，提醒管理员留意这台系统是否被用作攻击同网络的其他计算机的跳板。

May 15 16:27:20 host1 /kernel: fxp0: promiscuous mode enabled

虽然网络分析工具能将网络中传送的数据记录下来，但是网络中的数据流量相当大，如何对这些数据进行分析、分类统计、发现并报告错误却是更关键的问题。网络中的数据包属于不同的协议，而不同协议数据包的格式也不同。因此对捕获的数据进行解码，将包中的信息尽可能的展示出来，对于协议分析工具来讲更为重要。昂贵的商业分析工具的优势就在于它们能支持很多种类的应用层协议，而不仅仅只支持tcp、udp等低层协议。

从后面例子的tcpdump输出可以看出，tcpdump对截获的数据并没有进行彻底解码，数据包内的大部分内容是使用十六进制的形式直接打印输出的。显然这不利于分析网络故障，通常的解决办法是先使用带-w参数的tcpdump 截获数据并保存到文件中，然后再使用其他程序进行解码分析。当然也应该定义过滤规则，以避免捕获的数据包填满整个硬盘。

**4、表达式**

表达式(expression)可以用来在抓包的时候设置各种过滤条件等。expression是tcpdump最为有用的高级用法，可以利用它来匹配一些特殊的包。如果tcpdump中没有expression,那么tcpdump会把网卡上的所有数据包输出，否则会将被expression匹配的包输出。

expression 由一个或多个[primitives]组成，而[primitives]其实就是过滤条件，由一个或多个[qualitifer]加一个id(name)或数字组成，它们的结构如用正则表达式则可表示为：

expression = ([qualitifer］+(id|number))+

其中，[qualitifer］+(id|number)就是一个比较基本条件，qualitifer表达一些名称（项，变量），id或number则表示一个值（或常量）。

qualitifer共有三种，分别是：

(1)、type

表示id name或number涉及到的类型，如：host, net, port ,portrange等等。

例子：

"host foo" 此为一个简单的primitive，host为qualitifer, foo为id name，指定主机为foo。

"net 128.3" net为qualitifer, 128.3为number，指定网络为128.3。

"port 20" port为qualitifer, 20为number，指定端口为20。

每个privimtive必须有一个type词，如果表达式中没有，则默认是host.

(2)、dir

指定数据传输的方向，如：src, dst, src or dst, src and dst。

例子：

"dst net 128.3" 此为一个相对复杂的primitive，结构为"dir type number"表示的条件含义是：目标网络为128.3。

"src or dst port ftp-data" 此为比上一个相对简的结构，"src or dst"表示源或目标，"ftp-data"为id，表示ftp协议中数据传输端口，故整体表示源或目标端口ftp-data的数据包即匹配。

如果在一个primitive中没有dir词，此默认为src or dst. 如"host foo"则表示源或目标主机为foo的数据包都匹配。

(3)、proto

指定协议的关键字，主要包括fddi,ip,arp,rarp,tcp,udp等类型。

用来匹配某种特定的协议，包括：ether, fddi, tr, wlan, ip, ip6, arp, rarp, fddi,decnet,tcp和udp等。其实这些词经常用来匹配某种协议，是使用率最高的一组词了。Fddi指明是在FDDI(分布式光纤数据接口网络)上的特定的网络协议，fddi和ether具有类似的源地址和目的地址，所以可以将fddi协议包当作ether的包进行处理和分析。其他的几个关键字就是指明了监听的包的协议内容。如果没有指定任何协议，则tcpdump将会监听所有协议的信息包。ip icmp arp rarp 和 tcp、udp、icmp这些选项等都要放到第一个参数的位置，用来过滤数据报的类型。类似：tcpdump ip src……

例子：

tcp src port 80

ip dst host 192.168.1.1

　　通过上面介绍的三种qualitifer，我们很快就可以写出一个primitive。 如果出现type的话，一定会出现id或num;如果出现dir，那么也会出现type,如果不出现，默认为host; 而proto可单独出现，如 tcpdump 'tcp'。

除了这三种类型的关键字之外，其他重要的关键字如下：gateway, broadcast,less,greater,还有三种逻辑运算，取非运算是 'not ' '! ', 与运算是'and','&&';或运算 是'or' ,'││'；这些关键字可以组合起来构成强大的组合条件来满足人们的需要，后面举几个例子来说明。

**5、逻辑组合过滤条件**

通过学会写好每个 primtive，可以写出只有一个primtive的表达式，也可以把多个primitive组成一个expression，通过逻辑运算符连接起来就可以了，逻辑运算符有以下三个：

“&&” 或”and”

“||” 或“or”

“!” 或“not”

并且可通过()进行复杂的连接运算。如:

tcpdump 'ip && tcp’

tcpdump ‘ host 192.168.240.3 &&( tcp port 80 || tcp port 443)’

**6、高级过滤条件**

通过上面的各种primitive，我们可以写出很丰富的条件，如ip, tcp, udp,vlan等等。如IP，可以按址址进行匹，tcp/udp可以按端口匹配。但是，如果我想匹配更细的条件呢？如tcp中只含syn标志，fin标志的报文呢？上面的primitive恐怕无能为力了。不用怕，tcpdump为你提供最后一个功能最强大的primitive，记住是 primitive，而不是expression。你可以用多个这样的primitive组成更复杂的 expression。

若把这个形式记为A，那么你可这样写tcpdump 'A1 && A2 && ip src 192.168.200.1'，等等。下面我们就来分析A这个形式。

形式：expr relop expr

relop表示关系操作符，可以为>, < ,>=,<=, =, !=之一，

　　expr是一个算术表达式，由整数组成和二元运算符（＋，－，＊，/，＆，|, <<, >>)，长度操作，报文数据访问子。同时所有的整数都是无符号的，即0x80000000 和 0xffffffff > 0。

为了访问报文中的数据，可使用如下方式：

proto [ expr : size ]

proto表示该报文类型，expr的结果表示该报文的偏移，size为可选的，表示从expr偏移量起的szie个字节，整个表达式为proto报文 中,expr起的szie字节的内容（无符号整数）

例如：

'ether[0] & 1 !=0'：ether报文中第0个bit为1，即以太网广播或组播的primtive。

通过这种方式，我们可以对报文的任何一个字节进行匹配了，因此它的功能是十分强大的。

‘ip[0] = 4’：ip报文中的第一个字节为version，即匹配IPv4的报文，

'tcp[13] = 2'：匹配一个syn报文，因为tcp的标志位为TCP报文的第13个字节，而syn在这个字节的低1位，故匹配只有syn标志的报文,上述条件是可满要求的，并且比较严格。

'icmp[0]=8'：匹配ping命令的请求报文，因为icmp报文的第0字符表示类型，当类型值为8时表示为回显示请求。

**7、助记符号**

对于TCP和ICMP中常用的字节，如TCP中的标志位，ICMP中的类型，这个些偏移量有时会忘记。不过tcpdump为你提供更方便的用法，你不用记位这些数字，用字符就可以代替了。对于IP报文，没有提供字符支持，如果想匹配更细的条件，直接使用数字指字偏移量就可以了，不过要对IP报文有更深入的了解才可以。

对于ICMP报文，类型字节可以icmptype来表示它的偏称量，上面的primitive可改为'icmp[icmptype] =8'，如果8也记不住怎么办？tcpdump还为该字节的值也提供了字符表示，如'icmp[icmptype] = icmp-echo'。下面是tcpdump提供的字符偏移量：

icmptype：表示icmp报文中类型字节的偏移量

icmpcode:表示icmp报文中编码字节的偏移量

ICMP中类型字节的值可以是：

icmp-echoreply, icmp-unreach, icmp-sourcequench, icmp-redi﹔ect, icmp-echo, icmp-routeradvert, icmp-routersolicit,

icmp-timxceed, icmp-paramprob, icmp-tstamp, icmp-tstam﹑reply, icmp-ireq, icmp-ireqreply, icmp-maskreq, icmp-maskreply.

tcpflags表示TCP报文中标志位字节的偏移量

TCP中标志位字节的值可以是：

tcp-fin, tcp-syn, tcp-rst, tcp-push, tcp-ack, tcp-urg.

综上，我们可以写出下面的primitive:

'tcp[tcpflags] = tcp-syn':匹配只有syn标志设置为1的 tcp报文

'tcp[tcpflags] & (tcp-syn |tcp-ack |tcp-fin) !=0':匹配含有syn，或ack或fin标志位的TCP报文

**8、输出**

基本上tcpdump总的的输出格式为：系统时间 来源主机.端口 > 目标主机.端口 数据包参数

例如：

bash-2.02# tcpdump

tcpdump: listening on eth0

11:58:47.873028 202.102.245.40.netbios-ns > 202.102.245.127.netbios-ns: udp 50

11:58:47.974331 0:10:7b:8:3a:56 > 1:80:c2:0:0:0 802.1d ui/C len=43

0000 0000 0080 0000 1007 cf08 0900 0000

0e80 0000 902b 4695 0980 8701 0014 0002

000f 0000 902b 4695 0008 00

11:58:48.373134 0:0:e8:5b:6d:85 > Broadcast sap e0 ui/C len=97

ffff 0060 0004 ffff ffff ffff ffff ffff

0452 ffff ffff 0000 e85b 6d85 4008 0002

0640 4d41 5354 4552 5f57 4542 0000 0000

0000 00

^C

更多例子参见后面。

**[举例]**

\*监视第一个网络界面上所有流过的数据包:

root@quietheart:~/test#tcpdump

普通情况下，直接启动tcpdump将监视第一个网络界面上所有流过的数据包，注意这里使用超级用户。当用户上网得时候，就会将监视得数据打印出来。

\*重定向输出：

root@quietheart:~/test#tcpdump -l >tcpcap.txt

将得到的数据存入tcpcap.txt文件中。

\*匹配一次：

root@quietheart:~/test#tcpdump -c 1

这样就只匹配一个包，然后tcpdump就退出。

\*监视指定主机收发的数据包：

root@quietheart:~/test#tcpdump host 10.126.1.13

这样将会监视第一个网络接口上所有"10.126.1.13"主机收到的和发出的所有的数据包

\*监视指定主机和另外主机之间得通信：

root@quietheart:~/test#tcpdump host 10.126.1.222 and (10.126.1.1 or 10.126.1.13 )

这里将监视主机10.126.1.222和10.126.1.1或者10.126.1.13之间的通信，注意这里得括号。

\*监视时排除某一主机：

root@quietheart:~/test#tcpdump ip host 10.126.1.1 and ! 10.126.1.13

这里监视10.1.6.1.1除了和10.126.1.13之外所有主机的通信。

\*指定源主机和目的网络的过滤:

root@quietheart:~/test#tcpdump src host 10.126.1.222 and dst net 10.126.1.0/24

过滤的是源主机为10.126.1.222与目的网络为10.126.1.0/24的报头。

\*匹配指定网络：

root@quietheart:~/test# tcpdump -i eth1 net 10.1.10.0 mask 255.255.255.0

注意这里有掩码只支持IPV4，或者使用"tcpdump 10.1.10.0/24"，同样这个格式只对IPV4有用。

\*监视指定主机指定协议的包：

root@quietheart:~/test#tcpdump ip host 10.126.1.13

这里监视主机10.126.1.13收发的所有ip包。

\*监视指定主机和端口的数据包：

root@quietheart:~/test#tcpdump -i eth0 host 10.126.1.1 and port 80

这里会监视10.126.1.1端口80的TCP或UDP数据包，那么执行以下命令：

\*监视指定方向和接口的包1：

root@quietheart:~/test#tcpdump -i eth0 src host 10.126.1.222

这样会监视10.126.1.222发送的，eth0上面的包。

\*监视指定方向和接口的包2：

root@quietheart:~/test#tcpdump -i eth0 dst host 10.126.1.222

这样会监视10.126.1.222接收的，eth0上面的包。

\*监视指定链路层信息：

root@quietheart:~/test#tcpdump -b arp

这里，将只显示网络中的arp即地址转换协议信息。

\*过滤源主机mac为00:50:04:BA:9B的报头：

root@quietheart:~/test#tcpdump ether src 00:24:7e:03:62:14

这里，我们可以使用ifconfig查看本地主机的mac，使用arp -n来查看本地已知其他主机的mac。

\*过滤源主机为10.126.1.222目的主机端口不是telnet的报文：

root@quietheart:~/test#tcpdump src host 10.126.1.222 and dst port not telnet

\*匹配ether包：

root@quietheart:~/test#tcpdump -i eth1 'ether src 00:24:7e:03:62:14'

这里将会在eth1上面监听来自mac为"00:24:7e:03:62:14"的数据包。

\*监听ip广播组播数据包

root@quietheart:~/test#tcpdump 'ip dst 192.168.240.255'

这里，只需指明广播或组播地址即可。在同一个局域网的另一台主机运行"ping -b 192.168.240.255"即可发送ICMP的广播包。

\*监听80端口的tcp数据包：

root@quietheart:~/test#tcpdump 'tcp dst port 80'

这里，其实一般HTTP的通信数据，只需指定匹配端口为80的条件即可。

\*匹配ether广播包：

root@quietheart:~/test#tcpdump 'ether dst ff:ff:ff:ff:ff:ff'

这里，ether广播包的特征是mac全1。ether的组播包的特征是mac的最高位为1，其它位用来表示组播组编号，如果你想匹配其的多播组，知道它的组MAC地址即可。

\*匹配arp包：

root@quietheart:~/test#tcpdump arp

我们可以使用"arping <ip>"产生arp请求和接收答应报文，使用"tcpdump -p 'arp'"来监视，此处-p选项是使网络工作于正常模式（非混杂模式），这样是方便查看匹配结果，我们可以使用"ssh -p 80 <ip>"或者"wget <下载地址>"来产生http包。

\*匹配端口53上的udp数据包：

root@quietheart:~/test#tcpdump 'upd port 53'

实际53号端口是DNS协议所用的端口，可以用'ping -c 1 www.baidu.com'来产生DNS请求和答应。

\*指定范围的匹配条件：

root@quietheart:~/test#tcpdump portrange port1-port2

这里匹配端口在port1-port2范围内的ip/tcp，ip/upd，ip6/tcp和ip6/udp数据包。

\*抓取wireshark工具能够分析的包：

#tcpdump -w dumpfile.pcap

这样抓取内容存放在dumpfile.pcap中，可以用wireshark图形工具进行分析。有的说需要加-s参数，例如"tcpdump -s 0 -w dumpfile.pcap"。

**[其它]**

1、问题:

\*监视通过指定网关的数据包：

#tcpdump gateway hostname

注意实践发现这里gateway后面需要指定主机名称，ip好像不行。匹配使用hostname作为网关的数据包，即数据报中mac地址（源或目的）为hostname，但IP报的源和目的地址不是hostname的数据包。

\*截获指定端口和协议的包：

#tcpdump tcp port 23

这样会获取端口23上的tcp包（即telnet包）实践发现这里指定host会报错指定host之前加and就没有错误了即"tcpdump tcp port 23 and host 10.1.10.119"，但这样只能监视到本地到10.1.10.119上的telnet，不能其他主机到10.1.10.119的。

2、其它例子

下面给出用于tcpdump抓取报文的其它的条件，有待实践。

less length

匹配长度少于等于length的报文。

greater length

匹配长度大于等于length的报文。

ip protochain protocol

匹配ip报文中protocol字段值为protocol的报文

ip6 protochain protocol

匹配ipv6报文中protocol字段值为protocol的报文,如tcpdump 'ip protochain 6 匹配ipv4网络中的TCP报文，与tcpdump 'ip && tcp'用法一样，这里的&&连接两个primitive。6是TCP协议在IP报文中的编号。

ether broadcast

匹配以太网广播报文

ether multicast

匹配以太网多播报文

ip broadcast

匹配IPv4的广播报文。也即IP地址中主机号为全0或全1的IPv4报文。

ip multicast

匹配IPv4多播报文，也就是IP地址为多播地址的报文。

ip6 multicast

匹配IPv6多播报文，即IP地址为多播地址的报文。

vlan vlan\_id

匹配为vlan报文 ，且vlan号为vlan\_id的报文

host 192.168.240.91 && icmp[icmptype] = icmp-echo

192.168.240.91收发的icmp-echo类型的icmp报文。

host 192.168.1.100 && vrrp

192.168.1.100并且是vrrp协议。

ether src 00:00:00:00:00:02 && ether[0] & 1 !=0

发自00:00:00:00:00:02的广播或者组播的帧。

参考资料：

http://www.linuxso.com/command/tcpdump.html