网络抓包工具 tcpdump 使用指南

■ 北京 赵琳

编者按：tcpdump 是一款强大的网络抓包工具，它使用 libpcap 库来抓取网络数据包，这个库在几乎在所有的 Linux/Unix 中都有。熟悉 tcpdump 的使用能够帮助你分析调试网络数据。本文将通过一些具体的示例来介绍它在不同场景下的使用方法。

tcpdump 的常用参数：

```
$ tcpdump -i eth0 -nn -s0 -v port 80
```

-i：选择要捕获的接口，通常是以太网卡或无线网卡，也可以是 VLAN 或其他特殊接口。如果只有一个网络接口，则无需指定。

-nn：单个 n 表示不解析域名，直接显示 IP 地址；两个 n 表示不解析域名和端口。这样不仅方便查看 IP 地址和端口号，而且在抓取大量数据时非常高效，因为域名解析会降低抓取速度。

-s0：tcpdump 默认只会截取前 96 字节的内容，要想截取所有的报文内容，通过使用 -s number（number 是要截取的报文字节数），如果是 0 的话，表示截取报文全部内容。

-v：使用 -v、-vv 和 -vvv 来显示更多的详细信息，通常会显示更多与特定协议相关的信息。

port 80：表示仅抓取 80 端口上的流量，通常是 HTTP。

-p：不让网络接口进入混杂模式。默认情况下使用 tcpdump 抓包时，会让网络接口进入混杂模式。如果设备接入的交换机开启了混杂模式，使用 -p 选项可以有效地过滤噪声。

-e：显示数据链路层信息。默认情况下 tcpdump 不会显示数据链路层信息，使用 -e 选项可以显示源和目的 mac 地址，以及 VLAN tag 信息。

-w：用来把数据报文输出到文件。使用 tcpdump 截取数据报文的时候，默认会打印到屏幕的默认输出，你会看到按照顺序和格式，很多的数据一行行快速闪过，根本来不及看清楚所有的内容。

-A 表示使用 ASCII 字符串打印报文的全部数据，这样可以使读取更加简单，方便使用 grep 等工具解析输出内容。-X 表示同时使用十六进制和 ASCII 字符串打印报文的全部数据。这两个参数不能一起使用。例如：

```
$ tcpdump -A -s0 port 80
```

首先，抓取特定协议的数据。后面可以跟上协议名称来过滤特定协议的流量。以 UDP 为例，可以加上参数 udp 或 protocol 17，这两个命令意思相同。

```
$ tcpdump -i eth0 udp
$ tcpdump -i eth0 proto 17
```


(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

同理，tcp与protocol 6意思相同。

其次，行缓冲模式。如果想实时将抓取到的数据通过管道传递给其他工具来处理，需要使用-l选项来开启行缓冲模式(或使用-c选项来开启数据包缓冲模式)。使用-l选项可以将输出通过立即发送给其他命令，其他命令会立即响应。

$ tcpdump -I eth0 -s0 -l port 80 | grep 'Server:'

过滤器

把所有的数据截取下来，从里面找到想要的信息无疑是一件很费时费力的工作。而tcpdump提供了灵活的语法可以精确地截取关心的数据报，简化分析的工作量。这些选择数据包的语句就是过滤器(filter)。

1. Host过滤器

使用过滤器host可以抓取特定目的地和源IP地址的流量。例如：$ tcpdump -I eth0 host 10.10.1.1

该命令会抓取所有发往主机 10.10.1.1或者从主机10.10.1.1发出的流量。

也可以使用src或dst只抓取源或目的地。例如：$ tcpdump -i eth0 dst 10.10.1.1

该命令会只抓取发往主机 10.10.1.1的流量。

2. Network过滤器

Network过滤器用来过滤某个网段的数据，使用的是CIDR模式。可以使用四元组(x.x.x.x)、三元组(x.x.x)、二元组(x.x)和一元组(x)。四元组就是指定某个主机，三元组表示子网掩码为255.255.255.0，二元组表示子网掩码为255.255.0.0，一元组表示子网掩码为255.0.0.0。例如：$ tcpdump net 192.168.1

该命令会抓取所有发往网段192.168.1.x或从网段192.168.1.x发出的流量。

例如：$ tcpdump net 10

该命令会抓取所有发往网段10.x.x.x或从网段10.x.x.x发出的流量。

也可以使用src或dst只抓取源或目的地。

例如：$ tcpdump src net 10

也可以使用CIDR格式。例如：$ tcpdump src net 172.16.0.0/12

3. Proto过滤器

用来过滤某个协议的数据，关键字为proto，可省略。proto后面可以跟上协议号或协议名称，支持ICMP、IGMP、IGRP、PIM、AH、ESP、CARP、VRRP、UDP和TCP。

因为，通常的协议名称是保留字段，所以在与proto指令一起使用时，必须根据shell类型使用一个或两个反斜杠(/)来转义。Linux中的shell需要使用两个反斜杠来转义，MacOS只需要一个。

例如：$ tcpdump -n proto \\icmp# 或者 $ tcpdump -n icmp

该命令会抓取ICMP协议的报文。

4. Port过滤器

用来过滤通过某个端口的数据报文，关键字为port。例如：$ tcpdump port 389 该命令会抓取389端口的报文。

5. 组合过滤器

过滤的真正强大之处在于你可以随意组合它们，而连接它们的逻辑就是常用的与/AND/&&、或/OR/||和非/not/!。

and or &&

(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

or or ||
not or !

理解 tcpdump 的输出

截取数据只是第一步，第二步就是理解这些数据。下面就解释一下 tcpdump 命令输出各部分的意义。

实例：

21:27:06.995846 IP (tos 0x0, ttl 64, id 45646, offset 0, flags [DF], proto TCP (6), length 64)

192.168.1.106.56166 > 124.192.132.54.80: Flags [S], cksum 0xa730 (correct), seq 992042666, win 65535, options [mss 1460,nop,wscale 4,nop,nop,TS val 663433143 ecr 0,sackOK,eol], length 0

21:27:07.030487 IP (tos 0x0, ttl 51, id 0, offset 0, flags [DF], proto TCP (6), length 44)

124.192.132.54.80 > 192.168.1.106.56166: Flags [S.], cksum 0xedc0 (correct), seq 2147006684, ack 992042667, win 14600, options [mss 1440], length 0

21:27:07.030527 IP (tos 0x0, ttl 64, id 59119, offset 0, flags [DF], proto TCP (6), length 40)

192.168.1.106.56166 > 124.192.132.54.80: Flags [.], cksum 0x3e72 (correct), ack 2147006685, win 65535, length 0

最基本也是最重要的信息就是数据报的源地址／端口和目的地址／端口。上面的例子第一条数据报中，源地址IP是192.168.1.106，源端口是56166，目的地址是124.192.132.54，目的端口是80。“>”符号代表数据的方向。

此外，上面的三条数据是TCP协议的三次握手过程，第一条是SYN报文，通过Flags [S]可以看出。第二条是SYN报文的应答报文(SYN-ACK)，通过Flags [S.]可以看出。

工作实例

1. 提取HTTP用户代理

从HTTP请求头中提取HTTP用户代理：

```
$ tcpdump -nn -A -s1500 -l | grep "User-Agent:"
```

通过egrep可以同时提取用户代理和主机名(或其他头文件)：

```
$ tcpdump -nn -A -s1500 -l | egrep -i 'User-Agent:|Host:'
```

2. 只抓取HTTP GET和POST流量

抓取HTTP GET流量：

```
$ tcpdump -s 0 -A -vv 'tcp[((tcp[12:1] & 0xf0) >> 2):4] = 0x47455420'
```

也可以抓取HTTP POST请求流量：

```
$ tcpdump -s 0 -A -vv 'tcp[((tcp[12:1] & 0xf0) >> 2):4] = 0x504f5354'
```

注意：该方法不能保证抓取到HTTP POST有效数据流量，因为一个POST请求会被分割为多个TCP数据包。

上述两个表达式中的十六进制将会与GET和POST请求的ASCII字符串匹配。例如，tcp[((tcp[12:1] & 0xf0) >> 2):4] 首先会确定我们感兴趣的字节的位置(在TCP header之后)，然后选择我们希望匹配的4个字节。

3. 提取HTTP请求的URL

(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

提取 HTTP 请求的主机名和路径：

```
$ tcpdump -s 0 -v -n -l | egrep -i "POST /|GET /|Host:"
```

4. 提取 Cookies

提取 Set-Cookie（服务端的 Cookie）和 Cookie（客户端的 Cookie）：

```
$ tcpdump -nn -A -s0 -l | egrep -i 'Set-Cookie|Host:|Cookie:'
```

5. 抓取 ICMP 数据包

查看网络上的所有 ICMP 数据包：

```
$ tcpdump -n icmp
```

6. 抓取非 ECHO/REPLY 类型的 ICMP 数据包

通过排除 echo 和 reply 类型的数据包使抓取到的数据包不包括标准的 ping 包：

```
$ tcpdump 'icmp[icmptype] != icmp-echo and icmp[icmptype] != icmp-echoreply'
```

7. 抓取 SMTP/POP3 协议的邮件

可以提取电子邮件的正文和其他数据。例如，只提取电子邮件的收件人：

```
$ tcpdump -nn -l port 25 | grep -i 'MAIL FROM\|RCPT TO'
```

8. 抓取 NTP 服务的查询和响应

```
$ tcpdump dst port 123
```

9. 抓取 SNMP 服务的查询和响应

通过 SNMP 服务，网络运维人员可以获取大量的设备和系统信息。在这些信息中，系统信息最为关键，如操作系统版本、内核版本等。使用 SNMP 协议快速扫描程序 onesixtyone，可以看到目标系统的信息：

```
$ onesixtyone 10.10.1.10 public
```

可以通过 tcpdump 抓取 GetRequest 和 GetResponse：

```
$ tcpdump -n -s0 port 161 and udp
```

10. 切割 pcap 文件

当抓取大量数据并写入文件时，可以自动切割为多个大小相同的文件。例如，下面的命令表示每 3 600 s 创建一个新文件 capture-(hour).pcap，每个文件大小不超过 200*1 000 000 字节：

```
$ tcpdump -w /tmp/capture-%H.pcap -G 3600 -C 200
```

这些文件的命名为 capture-{1-24}.pcap，24 小时之后，之前的文件就会被覆盖。

11. 抓取 IPv6 流量

可以通过过滤器 ip6 来抓取 IPv6 流量，同时可以指定协议如 TCP：

```
$ tcpdump -nn ip6 proto 6
```

从之前保存的文件中读取 IPv6 UDP 数据报文：

```
$ tcpdump -nr ipv6-test.pcap ip6 proto 17
```

12. 检测端口扫描

在下面的例子中，你会发现抓取到的报文的源和目的一直不变，且带有标志位 [S] 和 [R]，它们与一系列看似随机的目标端口进行匹配。当发送 SYN 之后，如果目标主机的端口没有打开，就会返回一个 RESET。这是 Nmap 等端口扫描工具的标准做法。

```
$ tcpdump -nn
```

13. 过滤 Nmap NSE 脚本测试结果

本例中 Nmap NSE 测试脚本 http-enum.nse 用来检测 HTTP 服务的合法 URL。

在执行脚本测试的主机上：

```
$ nmap -p 80 —scrip
```


(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

t=http-enum.nse targetip

在目标主机上：

```
$ tcpdump -nn port 80 | grep "GET /"
```

14. 抓取DNS请求和响应

向Google公共DNS发起的出站DNS请求和A记录响应可以通过tcpdump抓取到：

```
$ tcpdump -I wlp58s0 -s0 port 53
```

15. 抓取HTTP有效数据包

抓取80端口的HTTP有效数据包，排除TCP连接建立过程的数据包（SYN/FIN/ACK）：

```
$ tcpdump 'tcp port 80 and (((ip[2:2] - ((ip[0]&0xf)<<2)) - ((tcp[12]&0xf0)>>2)) != 0)'
```

16. 找出发包最多的IP

找出一段时间内发包最多的IP，或者从一堆报文中找出发包最多的IP，可以使用下面的命令：

```
$ tcpdump -nnn -t -c 200 | cut -f 1,2,3,4 -d '.' | sort | uniq -c | sort -nr | head -n 20
```

cut -f 1,2,3,4 -d '.'：以.为分隔符，打印出每行的前四列。即IP地址。

sort | uniq -c：排序并计数

sort -nr：按照数值大小逆向排序

17. 抓取DHCP报文

抓取DHCP服务的请求和响应报文，67为DHCP端口，68为客户机端口。

```
$ tcpdump -v -n port 67 or 68
```

18. 将输出内容重定向到Wireshark

通常Wireshark（或tshark）比tcpdump更容易分析应用层协议。一般的做法是在远程服务器上先使用tcpdump抓取数据并写入文件，然后将文件拷贝到本地工作站上用Wireshark分析。

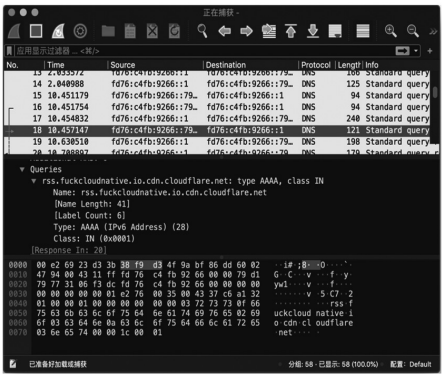

图1 抓取到的数据

还有一种更高效的方法，可以通过SSH连接将抓取到的数据实时发送给Wireshark进行分析。以MacOS系统为例，可以通过brew cask install wireshark来安装，然后通过下面的命令来分析：

```
$ ssh root@remotesystem 'tcpdump -s0 -c 1000 -nn -w - not port 22' | /Applications/Wireshark.app/Contents/MacOS/Wireshark -k -i -
```

例如，如果想分析DNS协议，可以使用下面的命令：

```
$ ssh root@remotesystem 'tcpdump -s0 -c 1000 -nn -w - port 53' | /Applications/Wireshark.app/Contents/MacOS/Wireshark -k -i -
```

抓取到的数据，如图1所示。

-c选项用来限制抓取数据的大小。如果不限制大小，就只能通过ctrl-c来停止抓取，这样一来不仅关闭了tcpdump，也关闭了wireshark。N

(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net