Table of Contents

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

Chapter 8. Tcpdump

8.1. Overview of Tcpdump

Most network administration tools are not based directly on the data being transmitted on a network, but rather on information related to that data. MRTG, for example, uses network bandwidth values. Other tools make use of system logs on network equipment or they test for system availability. It is sometimes necessary, however, to examine the packets themselves. Doing so will allow you to diagnose some particularly tricky network problems and can also serve as a hands-on approach to learning more about network protocols.

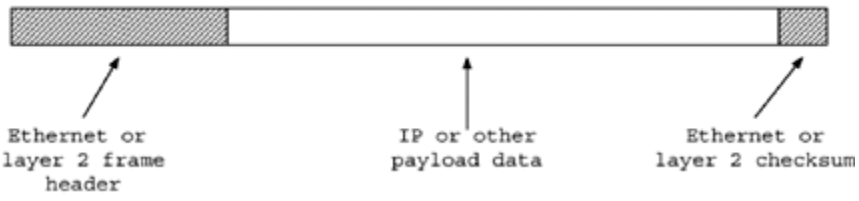
The most widely used open source tool for directly analyzing packets is a program called tcpdump, originally written by Van Jacobson. The standard tcpdump, through version 3.4, is maintained and distributed by the Lawrence Berkeley National Laboratory. Additional work has produced a second train of tcpdump releases as high as version 3.7.2 available from http://www.tcpdump.org/. The tcpdump that ships with most Linux distributions comes from this source. Both versions of tcpdump rely on the pcap library, a system for capturing packets across different operating systems. The pcap library is available from both the LBL and www.tcpdump.org.

One word of caution is necessary before you use tcpdump and other packet analyzers. Even though encryption is becoming more and more common in network protocols, there are still many protocols that transport data unencrypted. When using a packet analyzer to monitor network traffic, you will be able to view private data sent by users on the network—data that they may believe is not visible to others. There are serious legal implications to monitoring such data because it can be considered a form of wire tapping. Be sure to research relevant state and federal law before using a program such as tcpdump in an environment where user data will be present. When do you use a packet analyzer in this manner, remember to respect the privacy of other users as fully as possible and also ensure that you adhere to any privacy policies in place at your facility.

8.2. What Tcpdump Can Help You Do

Tcpdump will allow you to view the entire data portion of an Ethernet frame or other link layer protocol and can optionally print the frame header as well (see Figure 8.1). In common use this means tcpdump will allow you to view the entirety of an IP packet, an ARP packet, or any protocol at a higher layer than Ethernet. By default, tcpdump prints packets at the IP layer.

Figure 8.1. An Ethernet or Layer 2 Frame.



An example of typical tcpdump output looks like this:

```
11:51:46.637811 10.25.71.241.80 > 10.18.0.100.61965: . ack 415 ...
11:51:46.643077 10.25.71.241.80 > 10.18.0.100.61966: . ack 415 ...
```


Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

```
11:51:46.644830 10.209.29.151.80 > 10.18.0.100.61961: . ack 458...
11:51:46.653025 10.18.0.100 > 10.7.14.114: icmp: echo request (DF)
11:51:46.653226 10.7.14.114 > 10.18.0.100: icmp: echo reply (DF)
11:51:46.658675 10.209.29.137.53 > 10.18.0.100.53454: 46268*- 2...
11:51:46.659970 10.18.0.100.53454 > 10.70.10.79.53: 23134 A? sn...
11:52:24.306670 arp who-has 10.18.1.80 tell 10.18.0.1
```

Each line represents one packet. Details on how to read each field are presented later in the chapter, but at first glance, we can see an ARP request, a DNS query and response, and access to a web server.

In another mode, we can ask tcpdump to print all the data within each packet. The output is obviously much longer:

```
16:05:52.209620 10.7.21.77.80 > 10.18.0.100.62532: P 1:236(235)...
                        4500 0113 27a4 4000 3f06 d977 0a07 154d
                        0a12 0064 0050 f444 dec4 4cd8 5894 b1d4
                        5018 f82f c99a 0000 4854 5450 2f31 2e31
                        2033 3034 204e 6f74 204d 6f64 6966 6965
                        640d 0a44 6174 653a 2046 7269 2c20 3033
                        204a 616e 2032 3030 3320 3231 3a30 353a
                        3532 2047 4d54 0d0a 5365 7276 6572 3a20
                        4d49 5420 5765 6220 5365 7276 6572 2041
                        7061 6368 652f 312e 332e 3236 204d 6172
                        6b2f 312e 3420 2855 6e69 7829 206d 6f64
                        5f73 736c 2f32 2e38 2e39 204f 7065 6e53
                        534c 2f30 2e39 2e36 670d 0a43 6f6e 6e65
                        6374 696f 6e3a 204b 6565 702d 416c 6976
                        650d 0a4b 6565 702d 416c 6976 653a 2074
                        696d 656f 7574 3d31 352c 206d 6178 3d39
                        390d 0a45 5461 673a 2022 3236 3166 3932
                        6265 2d32 342d 3365 3135 6662 3164 220d
                        0a0d 0a
```

This is one entire IP packet, beginning with the IP version number (4) and the IP header length (5, representing the number of 32-bit words in the header).

The number of problems that can be solved with the help of tcpdump is limitless. Because it prints such detailed information about network traffic, tcpdump is to a network administrator what the microscope is to a biologist. It will not give you a feel for large trends as Neo or MRTG will, but it will give you a very clear picture of a specific part of your network. For this reason, it is an excellent tool to use when the problem is simply that something is not working properly.

Imagine a Web browser that is unable to load pages from a particular server; the Web browser just hangs. Is it a problem with the client, the server, or something in between? If you run tcpdump while loading the Web page, you can watch every stage of the transaction. You can make sure the DNS query for the Web server's hostname is completed, watch the client make the HTTP request to the server, and check to see if the server responds. Regardless of whether the server responds or not, you are now one step closer to understanding the problem.

Tcpdump can also help debug denial of service attacks. If a network is flooded and all other attempts to determine the source or destination of the traffic fail, tcpdump will show you the source address, destination address, and type of traffic involved. Even when other methods can pinpoint the traffic for you, tcpdump is often useful for examining the contents of the traffic should you wish to learn more about the nature of the attack.

There is one catch that can make tcpdump difficult to use: The machine running it must be connected to the network in such a way that it can view the traffic you wish to monitor. This means both that the machine must be connected to the same physical network as the one in question and that the physical network must allow your machine to view the traffic. Both issues are discussed in the section on using tcpdump.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

8.2.1. Limitations of Tcpdump

Although tcpdump will display very detailed information about the packets on a network, its view is in some ways limited by the network hardware. For example, a typical Ethernet card will discard packets with an invalid checksum. Therefore, tcpdump will not be a helpful tool for detecting this kind of broken packet on your network. For that, you will need specialized hardware.

Tcpdump is also able to report on only what it finds in the packet. If an IP address is forged in the packet, tcpdump has no ability to report anything else. Be aware that tcpdump is showing you only what the data is, not what it ought to be.

8.3. Installing Tcpdump

The sections that follow provide specifics on tcpdump installation.

8.3.1. You May Already Be a Winner

Modern Linux systems and some other operating systems now come with tcpdump already installed. If your Linux system has tcpdump installed, it can usually be found as `/usr/sbin/tcpdump`. If you do not know if your system has tcpdump installed, try logging in as root and typing:

```
Solaris# type tcpdump
```

If this returns "`tcpdump not found`," tcpdump probably is not installed on your system. You may also check for the existence of `/usr/local/bin/tcpdump` if it is possible another administrator installed the program before you. If you find tcpdump is already present on your system, you can skip the entire section on installing it.

Solaris does not come installed with tcpdump, but does come with a packet capturing program called snoop, installed as `/usr/sbin/snoop`. While snoop has a few features that tcpdump does not, it is to your advantage to install tcpdump as well. Tcpdump is widely used, and as a result, a number of programs can use its output to produce other reports. Tcpdump is also a better tool in some circumstances, including gathering packets over a long period of time.

8.3.2. Which Version to Build

As mentioned before, there are two trains of tcpdump software: the older and more standard version at the LBL and the newer version at http://www.tcpdump.org/. The latter version contains features that the older version does not, of course. You may choose to download and build either one; the installation process for both is fairly straightforward. In the following examples, the LBL version is used.

8.3.3. The Pcap Library

As mentioned earlier, tcpdump requires the pcap library, which can be downloaded from `ftp://ftp.ee.lbl.gov/libpcap.tar.Z`. On a Linux system, you will likely find that `/usr/lib/libpcap.a` or `/usr/lib/libpcap.so` already exists. If

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

so, you do not need to build the pcap library on your own. If you are on a system where it does not already exist, you will have to build it. Begin by uncompressing and unpackaging the file:

```
Solaris% uncompress libpcap.tar.Z
Solaris% tar xvf libpcap.tar
Solaris% cd libpcap-0.4
```

Then configure and build the package:

```
Solaris% ./configure
Solaris% make
```

When you are done, there will be a file named `libpcap.a` in the current directory. If you wish to install the pcap library on your system, you may do so by logging in to a root account and typing `make install`. However, you can also point the tcpdump build at the file you just created without installing it on your system.

8.3.4. Tcpdump

Retrieve the source for tcpdump from `ftp://ftp.ee.lbl.gov/tcpdump.tar.Z`. If you did not choose to install the pcap library on your system, you will want to place the tcpdump source so that its parent directory and the pcap source parent directory are the same. That is, from one directory you would like to see:

```
libpcap-0.4/  libpcap.tar   tcpdump.tar.Z
```

This will allow tcpdump to find the pcap library automatically. Now uncompress and unpackage the tcpdump source:

```
Solaris% uncompress tcpdump.tar.Z
Solaris% tar xvf tcpdump.tar
Solaris% cd tcpdump-3.4
```

Of course, the directory you change to will depend on the latest version number of tcpdump. Now build the package:

```
Solaris% ./configure
Solaris% make
```

And then you may install tcpdump from a root account:

```
Solaris# make install
Soalris# make install-man
```

The directory in which tcpdump is installed will depend on your system; on Solaris it will be `/usr/local/sbin`.

8.4. Using Tcpdump

Details on using tcpdump are presented in the following sections.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

8.4.1. Running as Root

Ordinarily, a network interface is not configured to capture every packet it sees on the network. It collects only packets that are addressed to that particular interface, or broadcast packets that are addressed to every interface.[1] In order to capture packets that are not addressed to the interface itself, tcpdump must put the interface into *promiscuous mode*. In promiscuous mode, all packets are collected regardless of their layer 2 destination address. On Unix-based operating systems, root privileges are required to put an interface into promiscuous mode; therefore, you will typically want to run tcpdump as root.[2] Occasionally, you may come across a version of tcpdump that requires a special flag to be set in order to enable promiscuous mode, but typically, tcpdump will attempt to enable it by default.

[1] Some Ethernet addresses are also available for "group" addressing. Packets addressed to one of these *multicast* addresses may be collected by a network card as well.

[2] Be aware that different systems deal with promiscuous mode differently. On some systems, for example, once an interface has been placed in promiscuous mode by the root account, other accounts will also have access to all packets. This is the exceptional case, however.

Do note that in certain extreme circumstances, enabling promiscuous mode can lead to degraded performance of the operating system. For example, a system with high-speed interfaces, or simply a very large number of interfaces in promiscuous mode, will place a heavy burden on the kernel. Under more typical conditions, such as a machine with one or two interfaces running 10 or 100Mb/s Ethernet, there should be little problem, however.

8.4.2. Command Line Options

Tcpdump has a number of command line options available, all of which are documented in the tcpdump man page. Some of the most common options are listed here.

As you experiment with the options below, note that your network topology may not allow you to view all of the traffic on your network. The reasons for this and possible solutions to this problem are described in detail in Section 8.4.7.

-n

By default, tcpdump performs a DNS query to look up the hostname associated with an IP address and uses the hostname in the output. For example:

```
12:54:07.594427 server.example.com.telnet > client.example.com...
12:54:07.686828 client.example.com.37580 > server.example.com...
```

Here, tcpdump read the source and destination IP addresses from the packet, looked up the hostnames associated with those addresses, and printed those names instead of the numeric IP addresses.

Though this is a convenient feature, it can have a serious impact on the performance of the program. If many different hosts are present, some with name servers on distant networks, tcpdump may experience delays while waiting for DNS queries to complete. Allowing tcpdump to look up hostnames is perfectly acceptable for short-term viewing when network conditions are favorable. But for long-term packet monitoring, or if you suspect tcpdump will have trouble performing DNS queries, it is preferable to disable hostname lookups. This is the function of the `-n` flag, as you can see below:

```
Linux# tcpdump -n
13:00:46.335152 10.18.0.100.23 > 10.56.0.43.37580: P 1:29(28) ...
13:00:46.435029 10.56.0.43.37580 > 10.18.0.100.23: . ack 29 ...
```

Also notice that instead of printing "telnet" as the port on the server, tcpdump used the numeric port number 23.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

-s *snaplen*

One counterintuitive default of tcpdump is that the amount of data captured is only the first 68 bytes of the packet. This is usually enough to grab the protocol headers, but it is not the entire packet. The *snaplen* option allows you to set the number of bytes tcpdump will grab from the packet. If you wish to view the entire packet (as with the `-x` option) or if you wish for the verbose options (`-v` and `-vv`) to have access to all of the data present in the packet, specify a snaplen size of 1500:

```
Linux# tcpdump -s 1500
```

We choose 1500 because it is the maximum size of the payload of an Ethernet frame. If we were using tcpdump on a network that is not Ethernet, we might need to set the snaplen size to an even larger value.

-x

The `-x` option instructs tcpdump to print the packet contents, which it does in hexadecimal notation:

```
13:11:44.459933 client.example.com.48630 > server.example.com...
                         4510 0028 7b8e 4000 fc06 dcc4 1265 0192
                         0a12 0064 bdf6 0017 b6e8 5b3c 2fdc c055
                         5010 210c a7f7 0000 0000 0000 0000
```

Note that if the snaplen, as described above, is smaller than the size of a packet, only the snaplen number of bytes will be printed in the output.

Later we will use a program to convert the hexadecimal output into a more readable format.

-v and -vv

As the previous examples have demonstrated, tcpdump understands some of the protocol information in the data it captures. In fact, it actually understands quite a bit more protocol information than it prints by default. If you add the `-v` option to the command line, tcpdump will print more information than usual about the protocols present, and if you instead use the `-vv` option, it will print even more detailed information. For example:

```
Linux# tcpdump -vv
...client.example.com.53454 > dns.example.com.domain: 15279 (38...
...dns.example.com.domain > client.example.com.53454: 15279* q:...
```

With the `-vv` option present, tcpdump now prints information about a DNS query being performed, including the name being looked up (server.example.com).

-q

The opposite of the `-v` and `-vv` options is the `-q` option, which instructs tcpdump to be more quiet; that is, to print less information on each line.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

-i *interface*

If your system has more than one interface, you can specify which one tcpdump should listen on with the `-i` option, as in:

```
Linux# tcpdump -i eth1
```

If you do not specify an interface, tcpdump will choose the lowest numbered interface that is up and is not the loopback interface.

-e

If the `-e` option is supplied on the command line, tcpdump will include the Ethernet (or other layer 2) header information in the output:

```
Linux# tcpdump -e
23:48:28.556873 0:3:ba:9:1f:36 0:5:dc:95:d0:a ip 76: client.exa...
```

The first hardware address (0:3:ba:9:1f:36) is the source Ethernet address, and the second is the destination address. The text "`ip`" indicates that the protocol is IP, and 76 is the length of the payload data.

-l

In some circumstances, you may wish to force tcpdump output to be line buffered. For example, if you are sending the output to a file but wish to view the results at the same time, run tcpdump as:

```
Linux# tcpdump -l | tee tcpdump.out
```

This will allow packets to be displayed as soon as tcpdump detects them, instead of waiting for a large amount of data to be present.

-w *file* and -r

In the preceding example, the output from tcpdump is stored directly in a file. While this is a reasonable way to capture and store data for later analysis, it is not very efficient and it can be difficult to work with because the format is not conducive to automated processing.

As an alternative, you can use the `-w` option to store packet data in a binary format:

```
Linux# tcpdump -w tcpdump.data
```

Tcpdump can later play back the data exactly as if it were being read from the wire, using the `-r` option:

```
Linux# tcpdump -r tcpdump.data
11:51:46.637811 10.25.71.241.80 > 10.18.0.100.61965: . ack 415...
11:51:46.643077 10.25.71.241.80 > 10.18.0.100.61966: . ack 415...
```

When you replay the data, you can change the options to tcpdump in order to view the data differently. There are also a number of programs available that can use the tcpdump data file format to process packets for other kinds of analysis.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

8.4.3. Filters

Everything on the tcpdump command line following the above options is an expression used to dictate exactly which packets should be captured and which should be ignored. Typically, you are interested in only a small number of the packets on the network. The filtering expression allows you to ignore anything you do not need to examine. A simple example is the best way to begin understanding how filters work:

```
Linux# tcpdump src client.example.com and dst server.example.com
```

In this example, tcpdump will print only those packets whose source address is that of client.example.com and whose destination address is that of server.example.com. The keywords `src` and `dst` are known as *primitives*. Another primitive is `host`, which specifies all traffic to or from a named host:

```
Linux# tcpdump host client.example.com
```

Here we view all traffic sent to or received from client.example.com. Some other useful tcpdump primitives are listed in Figure 8.2.

Figure 8.2. Some Tcpdump Packet Matching Primitives.

Primitive	**Function**
src *addr*	Source IP address matches *addr*
dst *addr*	Destination IP address matches *addr*
host *addr*	Source or destination IP address matches *addr*
ether <src/dst/host> *addr*	Ethernet address matches *addr*
[src/dst] net *net*	IP address is on network *net*
net *net*	Source or destination IP addr is on network *net*
net *net* mask *mask*	As above but network range defined by *mask*
[src/dst] port *port*	Port is *port*
port *port*	Source or destination port is *port*
less *octets*	Packet size is less than or equal to *octets*
greater *octets*	Packet size is greater than or equal to *octets*
icmp	Packet is an ICMP packet

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

Primitive	Function
tcp	Packet is a TCP packet
udp	Packet is a UDP packet
ip	Packet is an IP packet
arp	Packet is an ARP packet
broadcast	Packet is addressed to a broadcast address

Primitives can be combined with the boolean operators `and`, `or` and `not`, along with parentheses, to construct specialized filters. For example:

```
Linux# tcpdump "host client and not ( port telnet or port domain )"
```

will capture all packets sent to or from the host client.example.com but not those whose destination or source port is either telnet (port 25) or domain (port 53). We add the double quotes so that the parentheses will be passed directly to tcpdump instead of being interpreted by the shell.

8.4.4. Command Line Examples

Using the above knowledge, we can put together a number of useful tcpdump command lines. To display quick information on all traffic to or from the host broken.example.com:

```
Linux# tcpdump -q host broken.example.com
```

To view the entire packet for all bootp traffic:

```
Linux# tcpdump -xs 1500 port bootps or port bootpc
```

To leave tcpdump running for a long time, gathering data about ssh connections to client.example.com:

```
Linux# tcpdump -nxs 1500 -w tcpdump.data port 22 and host client
```

8.4.5. Understanding the Output

Some of the information printed by tcpdump is a bit cryptic, especially since the format is different for each protocol. The tcpdump man page lists the output format for each protocol, and the common ones are presented here as well.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

UDP Output Format

In the case of a simple UDP packet, the output format is:

time source > *destination*: udp *datalen*

So in the following line:

```
13:45:20.364930 10.7.15.82.2103 > 10.18.0.100.47028: udp 342 (DF)
```

we see that 10.7.15.82 on port 2103 sent 342 bytes to 10.18.0.100 on port 47028. The 342 bytes of data refers to the data portion of the UDP packet. The `(DF)` at the end indicates that the IP "don't fragment" bit is set.

TCP Output Format

For TCP packets, the output format is:

time source > *dest flags sequence* [ack *ack*] *win window* [*urgent*] [*options*]

For example:

```
...10.7.21.70.80 > 10.18.0.100.34639: P 1461:2921(1460) \
   ack 973 win 63268 (DF)
```

indicates that 10.7.21.70 on port 80 sent data to 10.18.0.100 on port 34639. The TCP PUSH flag was set, indicated by the "`P`." The string `1461:2921(1460)` gives us information about the TCP sequence number. It indicates that the packet is starting 1461 octets (eight-bit bytes) from the first sequence number tcpdump observed. This is called a relative sequence number. If you would rather view the actual sequence number used in the TCP packet, you can supply the `-S` argument on the tcpdump command line. The number after the colon is one more than the sequence number of the last byte in the packet, though this number is not really in the TCP header. The number in parentheses, 1460, is the length of the data sent.

The text "`ack 973`" indicates a TCP ACK was present and that the next expected sequence number in the other direction (data sent from 10.18.0.100 to 10.7.21.70) will be 973. This is also a relative sequence number if the `-S` flag is not used.

Finally, "`win 63268`" indicates that 10.7.21.70 will accept a TCP window size of 63268 octets. As in the UDP example, the `(DF)` represents the presence of the IP don't fragment option. In this example, the urgent TCP flag is not used and there are no extra options to report.

8.4.6. Viewing Packet Data

As described earlier, the `-x` option, when used in conjunction with the `-s 1500` setting, will instruct tcpdump to print the entire contents of a packet in hexadecimal. Because the hexadecimal output can be difficult to read, we can use an additional program to print character representations of each byte as well. Save the following into a file called `tpcdump-data-filter.pl`:

```
#!/usr/bin/perl
# This code is hereby placed in the public domain by its author,
# Marc Horowitz .  If you use it, it would be polite if you left
# my name on it, but there's no requirement.
$| = 1;
```


Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

```
while(<>) {
    if (/^\s/) {
        ($nospc = $_) =~ s/\s+//g;
        ($spc = $nospc) =~ s/(....)/$1 /g;
        ($bin = pack("H*",$nospc)) =~ tr/\000-\037\177-\377/./;
        printf("%16s%-45s%s\n","",$spc,$bin);
    } else {
        print;
    }
}
```

and give it execute permissions:

```
Linux# chmod u+x tcpdump-data-filter.pl
```

We can now pipe the tcpdump output through this program:

```
Linux# tcpdump -xls 1500 | ./tcpdump-data-filter.pl
tcpdump: listening on eth0
20:11:35.686269 host.example.com.53454 > c.gtld-servers.net.dom...
     4500 003d 9f2a 4000 ff11 add6 0a12 0064      E..=.*@........d
     c01a 5c1e d0ce 0035 0029 3674 8930 0000      ..\....5.)6t.0..
     0001 0000 0000 0000 0364 6e73 0765 7861      .........dns.exa
     6d70 6c65 0363 6f6d 0000 0100 01             mple.com.....
20:11:35.740531 host.example.com.34243 > web.example.com.80: P ...
     4500 03f4 a6f9 4000 4006 5641 0a12 0064      E.....@.@.VA...d
     0a07 154d 85c3 0050 f0b0 2504 41bc a72f      ...M...P..%.A../
     5018 60f4 3db0 0000 4745 5420 2f20 4854      P.'.=...GET / HT
     5450 2f31 2e30 0d0a 486f 7374 3a20 7765      TP/1.0..Host: we
     ...
```

The character representation does not add much meaning to the packet header data, but it makes it much easier to understand the the protocol data. In the first packet, we can see host.example.com perform a DNS query for dns.example.com. In the second packet, we can see host.example.com performing a “GET / HTTP/1.0” in an HTTP transaction with web.example.com.

8.4.7. Seeing It All

Before the arrival of the modern network switch, it was easy to view all of the traffic on an Ethernet network. Every packet on the network arrived at every network card, and as long as the card was in promiscuous mode, the operating system could capture every packet. On a switched network, this is no longer the case. Traffic patterns are optimized so that a link carries only the traffic destined for hosts connected to that link. [3] On a fully switched network this means tcpdump will be able to view only:

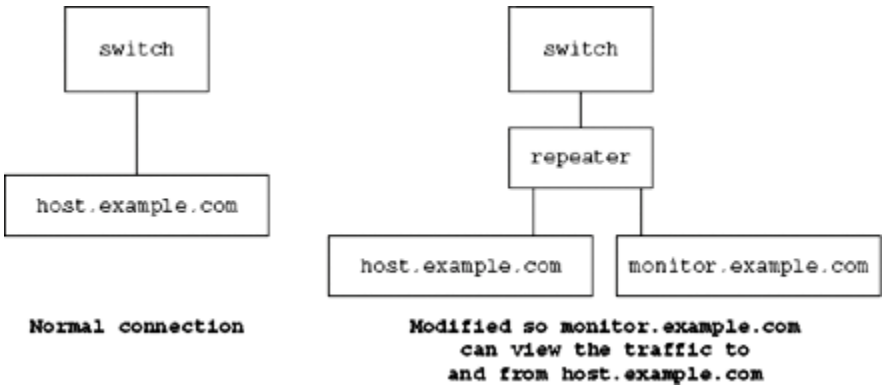
[3] Actually, this is not strictly true. Before a switch has figured out where a host resides, it sends traffic to every port. As a result, you may see occasional traffic for hosts on other links. This is one reason you should not rely on switching for data privacy.

- Traffic destined for your host
- Traffic originating from your host
- Broadcast traffic
- Small random amounts of traffic for other hosts (see the footnote)

This is a real setback if the point of using tcpdump is to help us monitor the packets sent by some other host. There are two ways to solve this problem. One is to connect the host in question and your monitoring host to a true repeater, as in Figure 8.3. This is a simple and effective solution if you can easily travel to the machine and attach another host appropriately. If not, an alternative solution is to configure your network hardware to forward the packets you are interested in to a port you can monitor them from. Not all network hardware is capable of doing this, however.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

Figure 8.3. Using a Repeater to Monitor Traffic.



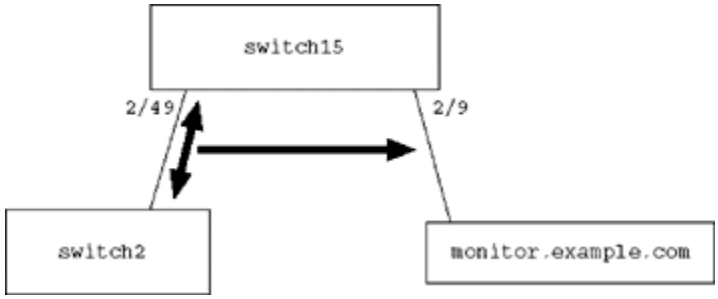
Monitoring on Cisco CatOS Devices

Cisco switches are capable of sending packets to additional ports for monitoring, though the syntax depends on which generation of switch software you are using. On the older CatOS systems, use the "`set span`" command (SPAN stands for switch port analyzer). From enable mode:

```
switch15> (enable) set span 2/49 2/9 both inpkts enable
```

In this example, all the traffic that would ordinarily be transmitted to port 2/49 or received from port 2/49 will also be sent to port 2/9. Now a host attached to port 2/9 can run tcpdump and monitor any packets that would be sent or received by a host attached to port 2/49 (see Figure 8.4).

Figure 8.4. Forwarding Traffic to an Additional Port for Monitoring.



On this particular switch, port 2/49 happens to be the uplink to the rest of the network, so monitoring its traffic allows us to monitor traffic of every device on the switch. Use caution when redirecting a large amount of traffic like this; if the destination link is not as large as the source link, you may flood the monitoring host.

The keyword `both` in the example above indicates that both transmitted and received traffic should be sent. The `inpkts enable` option is important; it tells the switch that it should process incoming packets to port 2/9 normally. The default behavior for a port with SPAN enabled is to ignore incoming packets. If you do not care about having your monitoring host able to talk to the rest of the network, you may leave the inpkts option out, but if you do wish to have the monitoring host accessible, be sure to include it. Note that some early versions of the CatOS software do not have the ability to use the `inpkts enable` option.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

The `set span` syntax also allows you to specify multiple source ports and ranges of ports. For example, `2/1-8` would represent port 2/1 through port 2/8. Using `2/1-8,2/10-50` would mean ports 2/1 through 2/8 and 2/10 through 2/50.

You can view the status of SPAN sessions with the `show span` command:

```
switch15> (enable) show span
Status          : enabled
Admin Source    : Port 2/49
Oper Source     : Port 2/49
Destination     : Port 2/9
Direction       : transmit/receive
Incoming Packets: enabled
```

Monitoring on Cisco IOS Devices

On Cisco devices running IOS (either switches or routers), port monitoring is configured with *monitoring sessions*. From configure mode:

```
switch18(config)#monitor session 1 source interface Gi1/1
switch18(config)#monitor session 1 destination interface Gi1/2
```

This would direct packets on port Gi1/1 to be copied to port Gi1/2. Remember to issue a `write mem` so that the configuration will still be in effect the next time the device is rebooted.

You can view monitoring sessions with the `show monitor` command:

```
switch18>show monitor

Session 1
---------
Source Ports:
    RX Only:       None
    TX Only:       None
    Both:          Gi1/1
Source VLANs:
    RX Only:       None
    TX Only:       None
    Both:          None
Destination Ports: Gi1/2
Filter VLANs:      None
```

8.5. Examples of Debugging with Tcpdump

The following sections provide specific examples of debugging with tcpdump.

8.5.1. Packet Flooding

Using tcpdump to find the source of a traffic flood is usually straightforward. Start by connecting a machine in a place where it will be able to monitor network traffic. If it is necessary to configure a switch so that packets will be sent to an additional port for monitoring, make sure to do so. Then run tcpdump and look for high talkers. Be sure to disable domain name lookups:

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

```
Linux# tcpdump -n
17:36:16.265220 10.255.255.27.1221 > 10.18.0.100.9995: udp 1168 (DF)
17:36:16.269171 10.255.255.27.1221 > 10.18.0.100.9995: udp 1168 (DF)
17:36:16.273130 10.255.255.23.1221 > 10.18.0.100.9995: udp 1168 (DF)
17:36:16.285228 10.255.255.27.1221 > 10.18.0.100.9995: udp 1168 (DF)
17:36:16.302173 10.255.255.27.1221 > 10.18.0.100.9995: udp 1168 (DF)
17:36:16.319372 10.255.255.27.1221 > 10.18.0.100.9995: udp 1168 (DF)
17:36:16.334600 10.7.15.65.7000 > 10.18.1.140.7001: rx ack (66) (DF)
17:36:16.334975 10.7.15.65.7000 > 10.18.1.140.7001: rx data (36) (DF)
17:36:16.336606 10.255.255.27.1221 > 10.18.0.100.9995: udp 1168 (DF)
17:36:16.336623 10.7.1.70.7000 > 10.18.1.140.7001: rx ack (66) (DF)
17:36:16.336939 10.7.1.70.7000 > 10.18.1.140.7001: rx data (36) (DF)
17:36:16.352253 10.255.255.27.1221 > 10.18.0.100.9995: udp 1168 (DF)
17:36:16.356199 10.255.255.27.1221 > 10.18.0.100.9995: udp 1168 (DF)
17:36:16.396921 10.255.255.27.1221 > 10.18.0.100.9995: udp 1168 (DF)
17:36:16.398427 10.155.0.153.57195 > 239.255.255.253.427: udp 49
17:36:16.400831 10.255.255.27.1221 > 10.18.0.100.9995: udp 1168 (DF)
17:36:16.404805 10.255.255.27.1221 > 10.18.0.100.9995: udp 1168 (DF)
17:36:16.408749 10.255.255.27.1221 > 10.18.0.100.9995: udp 1168 (DF)
17:36:16.412705 10.255.255.27.1221 > 10.18.0.100.9995: udp 1168 (DF)
17:36:16.416750 10.255.255.27.1221 > 10.18.0.100.9995: udp 1168 (DF)
```

From this short sample of output, we can see that there is a suspiciously large amount of traffic coming from 10.255.255.27 port 1221, directed at 10.18.0.100. Each packet is a UDP datagram with 1168 bytes of UDP payload data. The small difference in time stamps between packets helps convince us of the speed with which they are being sent.

Frequently, there is so much traffic on the network that it will not be so easy to determine who the high talker is. If that is the case, you may wish to rule out certain network traffic or include only certain network traffic in an attempt to focus in on the problem. For example, if you happen to know the flooding is directed at a particular host, use a filter to view traffic destined for that host:

```
Linux# tcpdump -n dst victim.example.com
```

8.5.2. A More Complicated Example

Imagine several workstations are having trouble accessing your Web server. The Web browser just hangs. Oddly, other machines on the same network have no problem reaching the server in a timely manner, and tests from your own workstation indicate there is no problem in connectivity. So you use a machine running tcpdump to examine the problem. First you may choose to look at the Web traffic originating from a workstation exhibiting the problem. You start tcpdump, instructing it to monitor port 80 (the port used for HTTP transactions) and then try to open the page in a Web browser:

```
Linux# tcpdump host client.example.com and port 80
```

But you see no traffic. Immediately, you can rule out the Web sever as the problem. If no traffic is sent to the Web server in the first place, the problem is probably not the fault of the server. So you decide to look at more traffic than just Web traffic by removing the port 80 restriction:

```
Linux# tcpdump host client.example.com
18:06:11.162372 client.example.com.45600 > dns.example.com.doma...
```

What is of interest is what you did *not* see. Although client.example.com makes a DNS request to dns.example.com, there is no response. If you look closer:

```
Linux# tcpdump -xls 1500 host client | ./tcpdump-data-filter.pl
18:14:12.842409 brokenclient.example.com.55313 > dns.example.co...
     4500 0048 058b 4000 ff11 9d80 0a12 0064      E..H..@........d
     0a05 061e d811 0035 0034 8a44 e4ca 0010      .......5.4.D....
     0001 0000 0000 0001 0377 7777 0765 7861      .........www.exa
     6d70 6c65 0363 6f6d 0000 0f00 0100 0029      mple.com.......)
     0800 0000 8000 0000                          ........
```


Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

In the body of the request, you can see the DNS lookup is for `www.example.com`. Now the problem is clear: The clients are attempting to look up the IP address of `www.example.com` before connecting to the Web server, but the DNS server is not responding. This explains why some workstations can connect without difficulty; they already have the IP address of the Web server cached from an earlier transaction.

8.6. Maintaining Tcpdump

Tcpdump requires essentially no maintenance. You may wish to upgrade the program on occasion, but it does not change very often. The most recent LBL version, 3.4, was released in 1998.

8.7. Other Packet Analyzers

Though tcpdump is the old, reliable standard for analyzing packet data, some newer tools offer attractive new features. For example, the Ethereal program, included with modern Linux distributions as `ethereal`, breaks down protocol data and displays it in a convenient graphical interface. It can capture live data or it can be run on a tcpdump datafile created with the `-w` option. This is a useful tool for digging deeply into the guts of a particular network protocol. More information on Ethereal is available at http://www.ethereal.com/.

Another popular tool is Snort, which is an intrusion detection system. Snort grabs data like tcpdump does but then analyzes it at a much higher level. It attempts to detect suspicious network traffic of all sorts, including various forms of attacks and probes. It is available from http://www.snort.org/.

8.8. References and Further Study

The man page installed with tcpdump includes information on features of the program not covered here. In particular, tcpdump understands a number of additional protocols, and the filter syntax is capable of more advanced expressions for specifying which packets should be captured.

RFC 791 describes the Internet Protocol (IP), including details of the header format. This explains the significance of the first 20 bytes viewed with the `-x` option. Further, the UDP protocol is described in RFC 768 and TCP, in RFC 793. ICMP is described in RFC 791. The books *Internetworking with TCP/IP* (Prentice Hall, 2000) by Douglas Comer and *TCP/IP Illustrated* (Addison-Wesley, 1994) by W. Richard Stevens both have descriptions and diagrams of all of these protocols, and both are easier reads than the RFCs.

There is a useful page of links to information about other packet analyzers at http://www.tcpdump.org/, under the Related Projects section. This includes pointers to programs such as Ethereal, TCPslice, and Snort.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003