# TCPDUMP

- Refer to book "Open Source Network Administration"
  - Online sample chapter: http://www.phptr.com/articles/article.asp?p=170902&seqNum=4
- Some tools are not based directly on the data being transmitted on a network, but information related to that data.
  - For example, network bandwidth values
  - System logs on network equipment
- Sometimes needs to examine the packets themselves.
  - Diagnose some particularly tricky network problems
- Widely used open source tool for directly analyzing packets: tcpdump
  - http://www.tcpdump.org/

# Caution

- Before you use tcpdump or other analyzer:
  - Will be able to see some private data
  - Consult/research Legal implication first
  - Respect the privacy of other users

## What Tcpdump can do for you

- View the entire data portion of an Ethernet frame or other link layer protocol
  - An IP packet
  - An ARP packet
  - Or any protocol at a higher layer than Ethernet.
- Very useful
  - Tcpdump is to a network administrator like a microscope to a biologist.
  - Give a very clear picture of a specific part of your network
  - Can be used when the problem is simply that something is not working properly.

## What tcpdump can do for you?

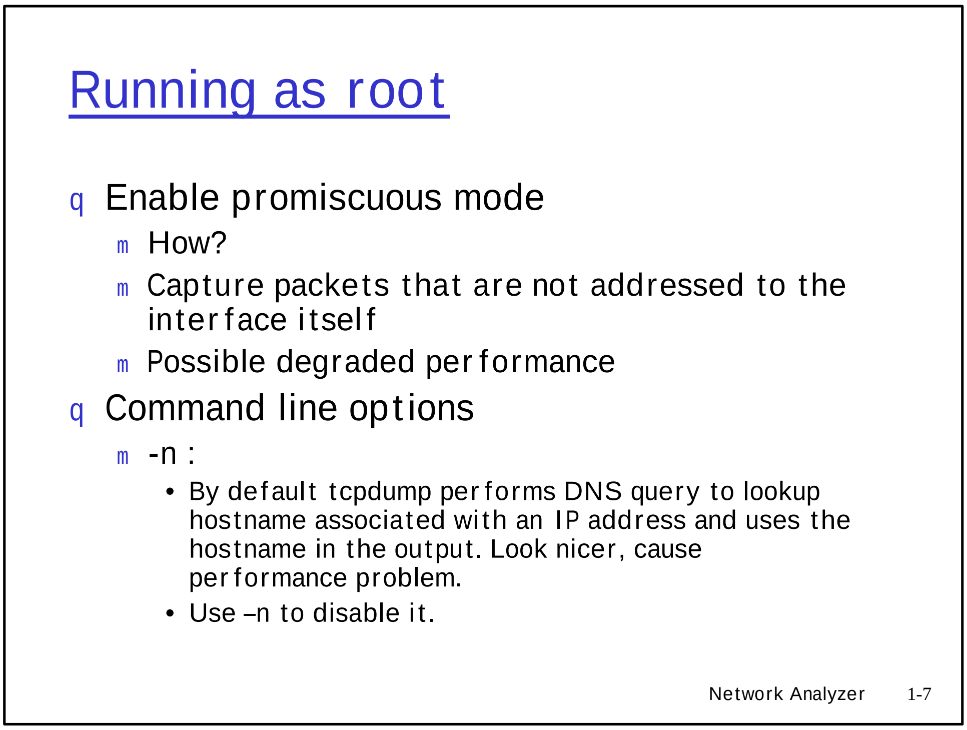
- Case1 : Web browser can not load pages from a server – it hangs.
  - Problem with client? Server? Or between?
  - Run tcpdump while loading
    - Watch every stage to see the following
      - DNS query
      - HTTP request to server
      - Server respond
- Case 2: help debug denial of service attacks.
  - Tcp show the source address, destination address, type of traffic, etc.
  - Check the packet contents to learn more about the nature of the attack.

## Limitations of Tcpdump

- Limited by network hardware
  - For example
    - Ethernet card will discard packets with an invalid checksum.
    - Tcpdump is not helpful for detecting this kind of broken packet on your network – need specialized hardware.
  - Tcpdump is showing you only what the data is, not what it ought to be -- has no ability to report "IP address is forged in the packet"

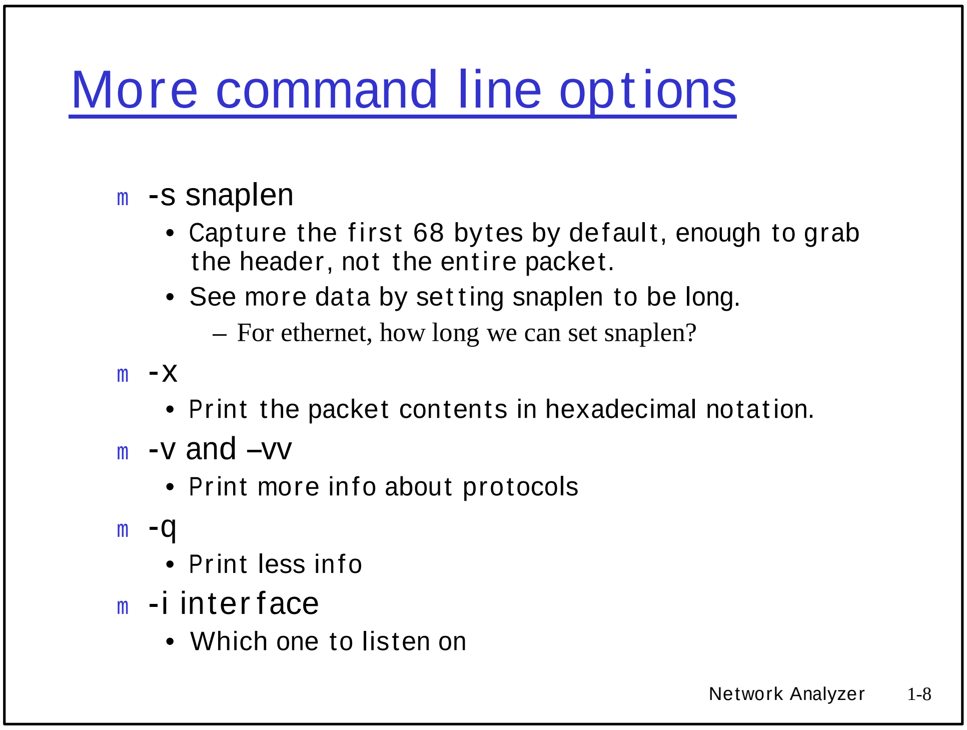
## Installing Tcpdump

- Already installed?
  - /usr/sbin/tcpdump
  - /usr/local/bin/tcpdump
  - Solaris comes with "snoop"
- Download software from http://www.tcpdump.org/
  - Down pcap library
- Download source code from ftp://ftp.ee.lbl.gove/tcpdump.tar.Z



## Running as root

- Enable promiscuous mode
  - How?
  - Capture packets that are not addressed to the interface itself
  - Possible degraded performance
- Command line options
  - -n :
    - By default tcpdump performs DNS query to lookup hostname associated with an IP address and uses the hostname in the output. Look nicer, cause performance problem.
    - Use –n to disable it.



## More command line options

- -s snaplen
  - Capture the first 68 bytes by default, enough to grab the header, not the entire packet.
  - See more data by setting snaplen to be long.
    - For ethernet, how long we can set snaplen?
- -x
  - Print the packet contents in hexadecimal notation.
- -v and –vv
  - Print more info about protocols
- -q
  - Print less info
- -i interface
  - Which one to listen on

# More command line options

- -e
  - Include Ethernet header
- -l
  - Force tcpdump output to be line buffered.
  - # tcpdump –l | tee tcpdump.out
- -w file and –r
  - Store the data in binary format and then play back as it were being read from the wire using –r

# Filters

- Following the command line options is the expression to dictate exactly which packets should be captured and which should be ignored.
- Primitive
  - src, dst
    - # tcpdump src client.example.com and dst server.example.com
      - Only those packet from “src” and to “dst”
  - host
    - # tcpdump host client.example.com
  - Can be combined with and, or and not with parentheses.
    - # tcpdump “host client and not (port telnet or port domain)”

## Some 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 |
| less octets | Packet size is less then or equal to octets |
| greater octets | Packet size is greater than or equal to octets |
| icmp | Packet is an ICMP packet |
| tcp | Packet is an TCP packet |
| udp | Packet is an udp packet |
| ip | Packet is an IP packet |
| arp | Packet is an ARP packet |
| broadcast | Packet is addresses to a broadcast address |

## Examples

- Use option and primitive, put together a number of useful tcpdump command line.
  - Display quick info on all traffic to/from a host
    - #tcpdump –q host broken.example.com
  - View entire packet for all bootp traffic
    - # tcpdump –xs 1500 port bootps or port bootpc
  - To gather ssh connections and leave tcpdump running for a long time to client.example.com
    - # tcpdump –nxs 1500 –w tcpdump.data port 22 and host client

## Understanding the output

- See Unix manpage
- UDP
  - Time source > destination: udp datalen
    - Example:
    - 13:45:20.364930 10.7.15.82.2103 > 10.18.0.100.47028: udp 342 (DF)
- TCP
  - Time source > dest flag sequence [ack ack] win window [urgent] [options]
    - Example:
    - ...10.7.21.70.80 > 10.18.0.100.34639: P 1461:2921(1460) ack 973 win 63268 (DF)

## Viewing Packet Data

- -x print out entire packets in hexadecimal.
- Use an additional program to print character representations of each byte.

```
#!/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.
  $| = 1;
  while(<>) {
     if (/^\s/) {
        ($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;
     }
  }
```

```
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
    …
```

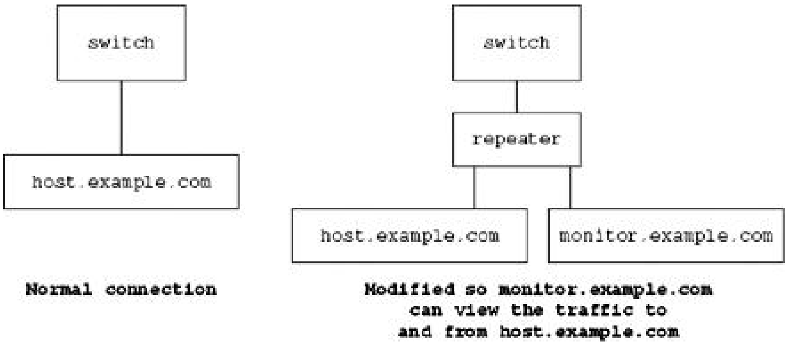
## Seeing It All

- Before modern network switch, is was easy to view all of the traffic on an Ethernet network.
- On a switched network, what will you see?

  tcpdump will be able to view only:
  - Traffic destined to your host
  - Traffic originated from you host
  - Broadcast traffic
  - Small random amounts of traffic for other hosts.

# Possible ways

- Connect the host in question and your monitoring host to a true repeater.



# Possible ways

- Configure network hardware to forward the packets you are interested in to a port you can monitor them from.
  - Not all hardware support.
  - Cisco switch is capable of doing so.

## Debugging with Tcpdump

- Packet flooding

```
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)
```

## Debugging with Tcpdump

- Webbrowser hangs
  - Step 1. Start tcpdump to monitor port 80
    - # tcpdump host client.example.com and port 80
      - No traffic, so web server is not a problem
  - Step 2. Start tcpdump for all
    - # tcpdump host client.example.com
      - Only DNS request, no response.

```
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
```