



# Wireshark 101

Ravi Boraskar

(Slides borrowed/stolen generously from all over the internet)

# Network Interfaces

- Linux box:
  - Show interfaces by “ifconfig”
  - Windows: “ipconfig /a” (I think)
- Look at routing table by running “netstat -r”
  - IP addresses are 32 bits
  - Network number, IP within the network
  - Next hop determined by longest prefix match on the IP address

# Wireshark

- World's most popular and complete protocol analyzer
- Wireshark is free software, and is available for Linux, Mac and Windows.
- More than 10 million downloads at SourceForge so far.

# What is wireshark?

- Wireshark is a **protocol analyzer**.
- This means Wireshark is designed to decode not only packet bits and bytes but also the relations between packets and protocols.
- Wireshark understands protocol sequences.

# What is Wireshark for YOU?

- Tool for examining packets on the ethernet/wireless mediums
- Need superuser access on machine
- Allows you to examine packets – all of them!
- Too much data, so you can employ filters
- Simplest case: just specify interface to snoop on

# Demo 1 – Basic Run

- Run wireshark on en1

home\_091115.pcap - Wireshark

File Edit View Go Capture Analyze Statistics Telephony Tools Help

Filter: ip.dst == 171.70.192.82 or arp

No.	Time	Source	Destination	Proto	Info
1	0.000000	192.168.1.104	171.70.192.82	UDP	NAT-keepalive
2	10.080090	192.168.1.104	171.70.192.82	UDP	NAT-keepalive
3	30.267573	IntelCor_51:d6:06	Broadcast	ARP	who has 192.168.1.1? Tell 192.168.1.104
4	30.269119	Cisco-Li_4b:0b:05	IntelCor_51:d6:06	ARP	192.168.1.1 is at 00:22:6b:4b:0b:05
5	30.269132	192.168.1.104	171.70.192.82	UDP	NAT-keepalive
6	40.363477	192.168.1.104	171.70.192.82	UDP	NAT-keepalive
7	60.728517	192.168.1.104	171.70.192.82	UDP	NAT-keepalive
8	70.822273	192.168.1.104	171.70.192.82	UDP	NAT-keepalive
9	131.29895	192.168.1.104	171.70.192.82	UDP	NAT-keepalive
10	252.37336	IntelCor_51:d6:06	Broadcast	ARP	who has 192.168.1.1? Tell 192.168.1.104
11	252.37496	Cisco-Li_4b:0b:05	IntelCor_51:d6:06	ARP	192.168.1.1 is at 00:22:6b:4b:0b:05
12	252.37498	192.168.1.104	171.70.192.82	UDP	NAT-keepalive
13	322.99476	192.168.1.104	171.70.192.82	UDP	NAT-keepalive
14	373.39786	192.168.1.104	171.70.192.82	UDP	NAT-keepalive
16	494.53741	IntelCor_51:d6:06	Broadcast	ARP	who has 192.168.1.1? Tell 192.168.1.104

Frame 20 (42 bytes on wire, 42 bytes captured)

Arrival Time: Nov 15, 2009 15:06:22.937349000  
 [Time delta from previous captured frame: 120.816474000 seconds]  
 [Time delta from previous displayed frame: 120.816474000 seconds]  
 [Time since reference or first frame: 675.690099000 seconds]  
 Frame Number: 20  
 Frame Length: 42 bytes  
 Capture Length: 42 bytes  
 [Frame is marked: False]  
 [Protocols in frame: eth:arp]  
 [Coloring Rule Name: ARP]  
 [Coloring Rule String: arp]

- Ethernet II, Src: IntelCor\_51:d6:06 (00:21:6a:51:d6:06), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  - Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  - Source: IntelCor\_51:d6:06 (00:21:6a:51:d6:06)
    - Type: ARP (0x0806)
- Address Resolution Protocol (request)

```

0000  ff ff ff ff ff ff 00 21  6a 51 d6 06 08 06 00 01  .....! jQ.....
0010  08 00 06 04 00 01 00 21  6a 51 d6 06 c0 a8 01 68  .....! jQ.....h
0020  00 00 00 00 00 00 c0 a8  01 01  ..... ..
  
```

File: "C:\Documents and Settings\anshah\My Doc... Packets: 30 Displayed: 29 Marked: 0 Time: 00:00:00.000 Profile: Default

# Wireshark Trace Example: What do these mean?

# Filters

- We are often not interested in all packets flowing through the network
- Use filters to capture only packets of interest to us
- Two kind of filters
  - Capture Filter: Filtered while capturing. Like TCPDump
  - Display Filter: More detailed filtering. Allows to compare values in packets. Not real time



# Demo 2

1. Capture only udp packets
  - Capture filter = “udp”
2. Capture only tcp packets
  - Capture filter = “tcp”

# Demo 2 (contd.)

1. Capture only UDP packets with destination port 53 (DNS requests)
  - “udp dst port 53”
2. Capture only UDP packets with source port 53 (DNS replies)
  - “udp src port 53”
3. Capture only UDP packets with source or destination port 53 (DNS requests and replies)
  - “udp port 53”

# Demo 2 (contd.)

1. Capture only packets destined to `www.cs.washington.edu`
  - “dst host `www.cs.washington.edu`”
2. Capture both DNS packets and TCP packets to/from [www.cs.washington.edu](http://www.cs.washington.edu)
  - “(tcp and host `www.cs.washington.edu`) or udp port 53”

# Display Filters

- Different Syntax
  - `frame.len > 10`
  - `ip.addr == 129.111.0.0/16` [CIDR masking]
- More expressive
  - `eth.src[1-2] == 00:83` [Check only bytes 1 and 2]
- Go crazy with logical expressions
  - `tcp.dst[0:3] == 0.6.29 xor udp.src[1] == 42`

# How to write filters

- Refer cheat sheet slides at the end of this presentation
- Refer the tcpdump man page and wireshark documentation

# Other tools

- TCPDump
  - Command line based [for the geeks in you!]
- IPsumdump
  - Summarize tcpdump output into human/machine readable form
  - <http://www.cs.ucla.edu/~kohler/ipsumdump/>
  - For instructions to use IPsumdump on EECS instructional accounts, see slide “Appendix: IPsumdump on EECS instructional accounts”
- Ethereal
  - ...is now wireshark
- Generally, wireshark is better!

# Security/Privacy Issues

- Wireshark allows you to monitor other people's traffic
- **WARNING: Do NOT use tcpdump to violate privacy or security**
- Use filtering to restrict packet analysis to only the traffic associated with your program. The following is one way to ensure that you see only traffic associated with your client:
  - `tcpdump -s 0 -r all_pkts.trace " -w my_pkts.trace "port 12345"`
  - where 12345 is the ephemeral port which your echo\_client uses to talk to the echo\_server.

# Cheat Sheet – Writing Filters (1)

- Specifying the hosts we are interested in
  - “dst host <name/IP>”
  - “src host <name/IP>”
  - “host <name/IP>” (either source or destination is name/IP)
- Specifying the ports we are interested in
  - “dst port <number>”
  - “src port <number>”
  - “port <number>”
  - Makes sense only for TCP and UDP packets



# Cheat Sheet – Writing Filters (2)

- Specifying ICMP packets
  - “icmp”
- Specifying UDP packets
  - “udp”
- Specifying TCP packets
  - “tcp”

# Cheat Sheet – Writing Filters (2)

- Combining filters
  - *and* (&&)
  - *or* (||)
  - *not* (!)
- Example:
  - All tcp packets which are not from or to host quasar.cs.berkeley.edu
    - tcpdump "tcp and ! host quasar.cs.berkeley.edu"*
  - Lots of examples in the EXAMPLES section of the man page