



Wireshark 101

Qiao Zhang
CSE 461 15sp Section #1

Slides adapted from
Ravi Bhorkar

What is Wireshark?

- Wireshark is a **network packet analyzer**
 - uses libpcap to capture packets
 - logs all packets seen by NIC
 - can display packet captured in real-time
 - can save packet trace as a file (*.pcap)
- Wireshark understands and decodes protocols
 - knows how packets are encapsulated
 - displays header in human-readable format
 - follows protocol sequence e.g. track a TCP stream

Why use Wireshark?

- Protocol analysis
 - verify correctness
 - analyze performance
 - better understanding of existing protocols
 - optimization and debugging of new protocols
- Works on Linux, OS X and Windows
 - works for both ethernet/wireless medium
- Has a GUI! Easier to use than tcpdump

display filter specification

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help



Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	128.208.2.151	74.125.127.104	TCP	66	56816 > http [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=1 SACK_PERM=1
2	0.008026	74.125.127.104	128.208.2.151	TCP	66	http > 56816 [SYN, ACK] Seq=0 Ack=1 Win=5720 Len=0 MSS=1430 SACK_PERM=1 WS=64
3	0.008058	128.208.2.151	74.125.127.104	TCP	54	56816 > http [ACK] Seq=1 Ack=1 Win=64240 Len=0
4	0.008552	128.208.2.151	74.125.127.104	HTTP	166	GET / HTTP/1.0
5	0.016630	74.125.127.104	128.208.2.151	TCP	60	http > 56816 [ACK] Seq=1 Ack=113 Win=5760 Len=0
6	0.025010	74.125.127.104	128.208.2.151	TCP	1484	[TCP segment of a reassembled PDU]
7	0.025101	74.125.127.104	128.208.2.151	TCP	1484	[TCP segment of a reassembled PDU]
8	0.025102	74.125.127.104	128.208.2.151	TCP	1282	[TCP segment of a reassembled PDU]
9	0.025103	74.125.127.104	128.208.2.151	TCP	1484	[TCP segment of a reassembled PDU]
10	0.025105	74.125.127.104	128.208.2.151	TCP	1484	[TCP segment of a reassembled PDU]
11	0.025106	74.125.127.104	128.208.2.151	TCP	1290	[TCP segment of a reassembled PDU]
12	0.025123	128.208.2.151	74.125.127.104	TCP	54	56816 > http [ACK] Seq=113 Ack=8185 Win=64240 Len=0

listing of captured packets

Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)

Encapsulation type: Ethernet (1)
Arrival Time: Feb 5, 2012 11:41:03.191994000 PST
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1328470863.191994000 seconds
[Time delta from previous captured frame: 0.000000000 seconds]
[Time delta from previous displayed frame: 0.000000000 seconds]
[Time since reference or first frame: 0.000000000 seconds]
Frame Number: 1
Frame Length: 66 bytes (528 bits)
Capture Length: 66 bytes (528 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ip:tcp]
Ethernet II, Src: Dell_d5:10:8b (00:25:64:d5:10:8b), Dst: IETF-VRRP-VRID_01 (00:00:5e:00:01:01)
Internet Protocol Version 4, Src: 128.208.2.151 (128.208.2.151), Dst: 74.125.127.104 (74.125.127.104)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
Total Length: 52
Identification: 0x5b3d (23357)
Flags: 0x02 (Don't Fragment)
Fragment offset: 0
Time to live: 128
Protocol: TCP (6)
Header checksum: 0x0000 [validation disabled]
Source: 128.208.2.151 (128.208.2.151)
Destination: 74.125.127.104 (74.125.127.104)
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
Transmission Control Protocol, Src Port: 56816 (56816), Dst Port: http (80), Seq: 0, Len: 0

details of selected packet header

```
0000 00 00 5e 00 01 01 00 25 64 d5 10 8b 08 00 45 00  .^...%d...E
0010 00 34 5b 3d 40 00 80 06 00 00 80 d0 02 97 4a 7d  4[=@...J]
0020 7f 68 dd f0 00 50 49 db 4d ff 00 00 00 00 80 02  h...PI..M...
0030 fa f0 4d 73 00 00 02 04 05 b4 01 03 03 00 01 01  .Ms.....
0040 04 02
```

packet content in hexadecimal and ASCII

Frame (frame), 66 bytes

Packets: 21 · Displayed: 21 (100.0%) · Load time: 0:00.086

Profile: Default

Network Interfaces

- Need to specify one for Wireshark to snoop on
- Show network interfaces:
 - On a linux box: “ifconfig”
 - Windows: “ipconfig /a”
 - Wireshark menu: Capture->Interfaces
- Must select loopback interface (lo0) to see packets from your own machine to itself e.g. “ping localhost”

Demo 1 – Basic Run

- Run wireshark on en1

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

- Capture only udp packets
 - Capture filter = “udp”
- Capture only tcp packets
 - Capture filter = “tcp”

Demo 2 (contd.)

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

Demo 2 (contd.)

- Capture only packets destined to www.cs.washington.edu
 - “dst host www.cs.washington.edu”
- Capture both DNS packets and TCP packets to/from 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`
- Cheat sheet

http://packetlife.net/media/library/13/Wireshark_Display_Filters.pdf

How to write filters

- Refer cheat sheet slides at the end of this presentation
- Refer the tcpdump man page and wireshark documentation
 - capture filters
<https://wiki.wireshark.org/CaptureFilters>
 - display filters
<https://wiki.wireshark.org/DisplayFilters>

Security/Privacy Issues

- Wireshark allows you to monitor other people's traffic
- **WARNING: Do NOT use wireshark to violate privacy or security**
- Use filtering to restrict packet analysis to only the traffic associated with your program
 - filter based on port that your application uses

Thank You

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:
 - *“tcp and ! host quasar.cs.berkeley.edu”*
 - All tcp packets which are not from or to host quas