

# Network “sniffing”— packet capture and analysis

October 5, 2012

## Administrative – submittal instructions

- answer the lab assignment’s 13 questions in numbered list form, in a Word document file. (13<sup>th</sup> response is to embed a screenshot graphic.)
- email to [csci5301@usc.edu](mailto:csci5301@usc.edu)
- exact subject title must be “snifflab”
- deadline is start of your lab session the following week
- reports not accepted (zero for lab) if
  - late
  - you did not attend the lab (except DEN or prior arrangement)
  - email subject title deviates

## Administrative – free week

- no lab next Wednesday nor Friday Oct 10 and 12
- no lecture next Friday Oct 12
- this lab performed Oct 17 and 19
- Next lecture Oct 19
- do your best on on midterm!

### Calendar schedule

<u>Fri lecture</u>	<u>Topic</u>	<u>lab meeting week of:</u>	<u>due:</u>
9/7	Cryptography	9/10	
9/14	Authentication	9/17	
9/21	Authorization	9/24	
9/28	Application security	10/1	10/19
10/5	Packet sniffing		
	- 10/12 Midterm -		
	Packet sniffing	10/15	10/26 Fri
10/19	Firewalls	(remote)	10/29 Mon
10/26	Intrusion detection	10/29	11/9 Fri
11/2	arp spoofing	(remote)	11/12 Mon
11/9	Tunnels and VPNs	(remote)	11/19 Mon
11/16	Computer forensics	(remote)	11/30 Fri

## DETER preparations

- coming soon
  - next lecture topic will be done on DETER
  - so will 3 of the 4 remaining thereafter
- you have an account
  - created a while ago
  - you received an advisory email at that time
- to-do for you - if not already done
  - make profile changes requested per your advisory email
  - do the “get your feet wet” exercise, per web page
  - review <https://education.deterlab.net/DETERintro/DETERintro.html#access>

## DETER activity timing

- machine quotas in place for our whole class
  - software enforced by DETER
  - timed to match our assignment schedule
- formal reservation intervals in DETER's schedule database per next slide
- assignments to be done during those intervals
- due dates per earlier slide or class web page

## DETER calendar (4 labs)

<u>Fri lecture</u>	<u>Topic</u>	<u>interval to do lab*</u>
10/5	Packet sniffing	non-deter
→ 10/19	Firewalls	10/20 – 10/26 (inclusive)
10/26	Intrusion detection	non-deter
→ 11/2	arp spoofing	11/3 – 11/9
→ 11/9	Tunnels and VPNs	11/10 – 11/16
→ 11/16	Computer forensics	11/17 – 11/23

\* when we have machine reservations in place with DETER administration

## Packet sniffer

- A tool that captures, interprets, and stores network packets for analysis
- also known as
  - network sniffer
  - network monitor
  - packet capture utility
  - protocol analyzer
- is intimately “network-y”

## Sniffing in security context

an introductory counterpoint

- conventional wisdom
  - “hacking” is emblematic of popular security talk
  - and is all about the outside menace
  - popular conclusion: “security is about networks”
- reality
  - the outside is there
    - but don't forget
  - the inside too!! does “security” vanish when net cable unplugged?

## Half of security *unrelated* to nets

- purely local dimensions
  - physical security
  - BIOS/bootloader security
  - filesystem permissions
  - execution jails
  - encrypted filesystems
  - etc
- network aspects
  - packet sniffing
  - remote backup and logging
  - port scanning
  - tunnels

but  
**today's topic**  
is in the network category



## Wireshark product background

- principal author Gerald Combs
- original name “ethereal” (changed 2006, legal reasons)
- open source
- equivalent linux and Windows versions

## Related software

- pcap
  - the underlying library
  - pcap captures the packets
  - Wireshark displays them (graphically)
- tcpdump
  - rides on pcap like Wireshark
  - displays what pcap captures (character mode)
  - very widespread
- others
  - tshark, character mode version in Wireshark's stable
  - Network Monitor - Microsoft
  - snoop - Sun Microsystems
  - ettercap
  - snort

## netcat product background

- a “general purpose” client and server
- there's more than one (hobbit's, GNU's)
  - different authors
  - different features
  - different syntax
- cryptcat
  - adds filestream en/de-cryption
- for you to generate something to send a server in this exercise

## ssh – secure shell

- creates an encrypted network conversation
- for you to compare with an unencrypted one in this exercise
- by capturing both

## Foundation concept: frames

- are what Wireshark is for capturing
- a.k.a. packets, datagrams, segments, protocol data units
- they come in nested groups

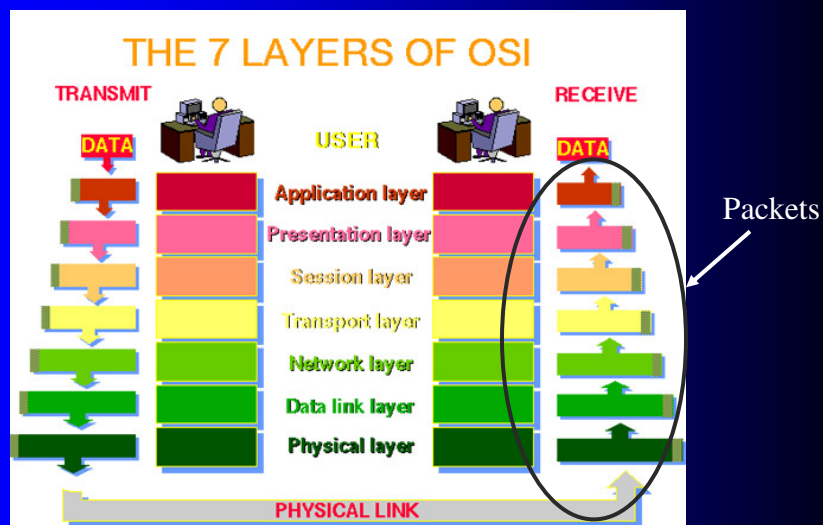
# Nesting / successive enveloping



Russian laquer dolls

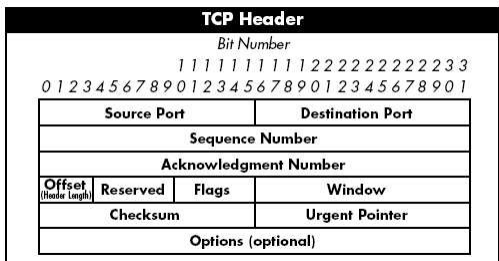
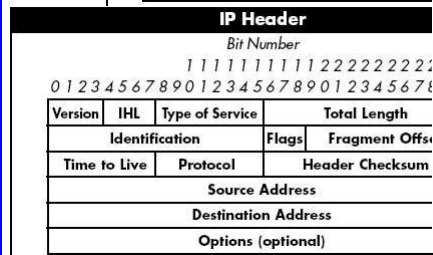
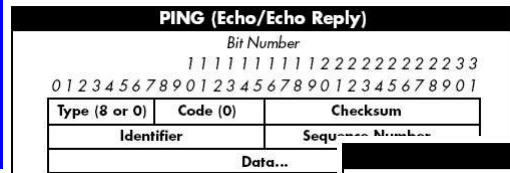
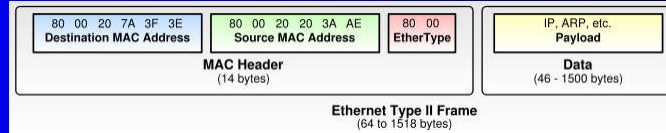


# How data gets enveloped

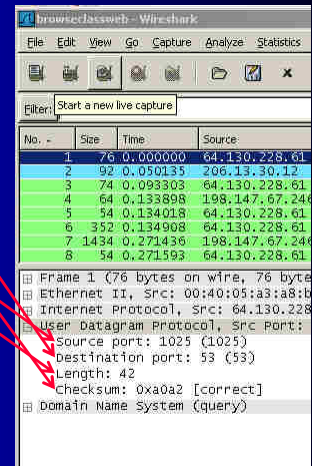
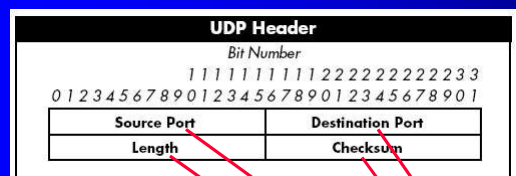




# Packets have detailed structure



# Packets have detailed structure



- Wireshark knows the structures
- for 800-900 protocols
- turns byte dump into intelligible decode, in the details pane

# Wireshark interface components

packet list pane →

packet details pane →

packet bytes pane →

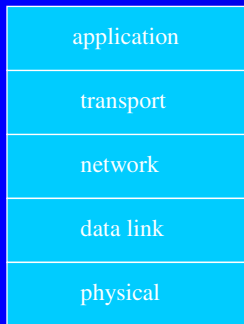
The screenshot shows the Wireshark interface with three panes. The top pane is the packet list, showing a list of captured packets. The middle pane is the packet details pane, showing the hierarchical structure of the selected packet (packet 6). The bottom pane is the packet bytes pane, showing the raw hexadecimal and ASCII data of the selected packet. Red boxes highlight the details and bytes panes.

No.	Size	Time	Source	Destination	Protocol	Info
1	76	0.000000	64.130.228.61	206.13.30.12	DNS	Standard query A homepage.smc.edu
2	92	0.050135	206.13.30.12	64.130.228.61	DNS	Standard query response A 198.147.67.246
3	74	0.093303	64.130.228.61	198.147.67.246	TCP	1195 → 80 [SYN] Seq=0 Len=0 MSS=2460 TSV=4124830 T...
4	64	0.133898	198.147.67.246	64.130.228.61	TCP	80 → 1195 [SYN, ACK] Seq=0 Ack=1 Win=8280 Len=0 MS...
5	54	0.134018	64.130.228.61	198.147.67.246	TCP	1195 → 80 [ACK] Seq=1 Ack=1 Win=32120 Len=0
6	352	0.271903	64.130.228.61	198.147.67.246	HTTP	GET /morgan_david/ HTTP/1.0\r\n\r\nConnection: keep-alive\r\nUser-Agent: Mozilla/4.0 [en] (X11; Linux 2.2.12-20 i686)\r\nHost: homepage.smc.edu\r\nAccept: image/gif, image/x-bitmap, image/jpeg, image/pjpeg, image/png, */*\r\nAccept-Encoding: gzip\r\nAccept-Language: en\r\nAccept-Charset: iso-8859-1, *, utf-8\r\n\r\n
7	1434	0.271436	198.147.67.246	64.130.228.61	TCP	[TCP segment of a reassembled PDU]
8	54	0.271593	64.130.228.61	198.147.67.246	TCP	1195 → 80 [ACK] Seq=299 Ack=1381 Win=31740 Len=0

packet 6's details

packet 6's bytes

# Stack correlation



highest-layer protocol that each packet contains

The screenshot shows the Wireshark interface with the packet details pane expanded to show the protocol stack for packet 6. Red arrows point from the protocol names in the details pane to the corresponding layers in the OSI model diagram. The protocols shown are Hypertext Transfer Protocol (application), Transmission Control Protocol (transport), Internet Protocol (network), Ethernet II (data link), and Ethernet II (physical).

## Wireshark taps interfaces

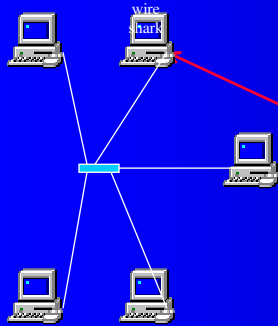
- probe takes measurement “where it is”
- sees whatever is at the interface (e.g, NIC)
- sees nothing else
- does not see “what’s on the network”
- limits value on host connected to a switch (versus a hub)

It's 70° in L.A.



No, it's 70°  
*right here*

There's a port scan on the network



No, there's a port scan  
*right here*

## Two what-to-capture restrictions

- Involuntary: can't capture what doesn't appear on the interface in the first place
- Voluntary: packet filter expressions

## Packet filter expressions using address primitives

- host 200.2.2.1
- src host 200.2.2.2
- dst host 200.2.2.2
- 'ip[16]>=224'
- 'ip[2:2]>512'
- 'ether[0]&1=1'

## Packet filter expressions using protocol primitives

- ip
- tcp
- udp
- icmp

## Booleans

- and
- or
- not

## 2 different filters, 2 different syntaxes

- capture filters (during capture)
  - shares same syntax as tcpdump uses
- display filters (after the fact)
  - Wireshark's own syntax
  - can auto-generate filter expression from model packet

## These syntaxes semantically same

enter display filter here while displaying

enter capture filter here before capturing

No.	Time	Source	Destination	Protocol	Info
13	0.0018309	192.168.3.28	69.242.106.5	TCP	2098 → 2601 [SYN] Seq=0 Len=0 MSS=1360
14	0.0018433	192.168.3.28	69.242.106.5	TCP	2098 → 2601 [SYN] Seq=0 Len=0 MSS=1360
25	0.1019133	69.242.106.5	192.168.3.28	TCP	2601 → 2098 [SYN, ACK] Seq=0, Ack=1, Win=65535, Len=0 MSS=1360
26	0.1020009	192.168.3.28	69.242.106.5	TCP	2098 → 2601 [ACK] Seq=1, Ack=1, Win=17680, Len=0
27	0.1021159	192.168.3.28	69.242.106.5	TCP	2098 → 2601 [ACK] Seq=1, Ack=1, Win=17680, Len=0
28	0.102938	192.168.3.28	69.242.106.5	TCP	2098 → 2601 [ACK] Seq=1, Ack=1, Win=17680, Len=0
29	0.103041	192.168.3.28	69.242.106.5	TCP	2098 → 2601 [ACK] Seq=1, Ack=1, Win=17680, Len=0
33	0.225847	69.242.106.5	192.168.3.28	TCP	2601 → 2098 [ACK] Seq=1, Ack=1, Win=17680, Len=0
34	0.227541	192.168.3.28	69.242.106.5	TCP	2098 → 2601 [ACK] Seq=1, Ack=1, Win=17680, Len=0
35	0.227631	192.168.3.28	69.242.106.5	TCP	2098 → 2601 [ACK] Seq=1, Ack=1, Win=17680, Len=0
36	0.228930	69.242.106.5	192.168.3.28	TCP	2601 → 2098 [ACK] Seq=1, Ack=1, Win=17680, Len=0
37	0.228990	192.168.3.28	69.242.106.5	TCP	2098 → 2601 [ACK] Seq=1, Ack=1, Win=17680, Len=0
38	0.229092	192.168.3.28	69.242.106.5	TCP	2098 → 2601 [ACK] Seq=1, Ack=1, Win=17680, Len=0
40	0.312892	69.242.106.5	192.168.3.28	TCP	2601 → 2098 [ACK] Seq=1, Ack=1, Win=17680, Len=0

## If you want to see network traffic besides your own

- make sure NIC is in promiscuous mode
- operate in a network with a hub, not a switch
  - not your choice if you're not net admin
- use a switch with a management port that receives all traffic
- sniff by remote access on computers at other places in the network, save the capture to a file, transfer the file to Wireshark

## info

- <http://www.wireshark.org/>
- <http://wiki.wireshark.org/>
- “Packet Sniffing In a Switched Environment”
  - [http://www.sans.org/reading\\_room/whitepapers/networkdevs/244.php](http://www.sans.org/reading_room/whitepapers/networkdevs/244.php)