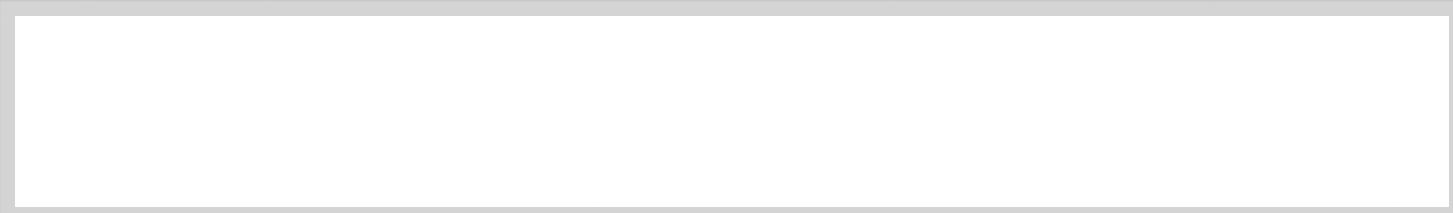


# USING WIRESHARK TO CAPTURE AND ANALYZE NETWORK DATA

CPSC 441 TUTORIAL – JANUARY 30, 2012  
TA: RUITING ZHOU

The content of these slides are taken from CPSC 526 TUTORIAL by Nashd Safa  
(Extended and partially modified)

- 
- TA: Ruiting Zhou
  - Email: [rzho@ucalgary.ca](mailto:rzho@ucalgary.ca)
  - CT hour: Wednesday 4:00pm-5:00pm  
Friday: 11:00am-12:00am
  - Math Science Building
  - 1<sup>st</sup> floor, Computer science Lab

# WIRESHARK

- **Wireshark** (Originally named Ethereal) is a free and open-source packet analyzer
- It is used for network troubleshooting, analysis, software and communication protocol development, and education.
- It has a graphical front-end, and many more information sorting and filtering options.

## FEATURES AND FUNCTIONALITIES OF WIRESHARK

- **Wireshark** is software that "understands" the structure of different **networking protocols**. Thus, it is able to display the encapsulation and the fields along with their meanings of different packets specified by different networking protocols.
- Live data can be read from a number of types of network, including Ethernet, IEEE 802.11, PPP...
- Data display can be refined using a display filter.

# INSTALLING WIRESHARK

- Download Wireshark from <http://www.wireshark.org/download.html>
- Choose appropriate version according to your operating system
- (For Windows), during installation agree to install **winpcap** as well.
- **pcap** (packet capture) consists of an application programming interface (API) for capturing network traffic. Unix-like systems implement pcap in the libpcap library. Windows uses a port of libpcap known as **WinPcap**.
- <http://wiki.wireshark.org/CaptureSetup>  
Provides a good tutorial on how to capture data using WireShark

## BEFORE CAPTURING DATA

- **Are you allowed to do this?**
- Ensure that you have the permission to capture packets from the network you are connected with. (Corporate policies or applicable law might prohibit capturing data from the network)
- **General Setup**
- Operating system must support packet capturing, e.g. capture support is enabled
- You must have sufficient privileges to capture packets, e.g. root / Administrator privileges
- Your computer's time and time zone settings should be correct

# CAPTURING DATA

- Check the interfaces are correctly listed

The screenshot displays the Wireshark Network Analyzer interface. The title bar reads "The Wireshark Network Analyzer [Wireshark 1.6.5 (SVN Rev 40429 from /trunk-1.6)]". The menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Tools, Internals, and Help. The toolbar contains various icons for file operations and analysis. A filter bar is visible with the text "Filter: Expression... Clear Apply".

The main interface is divided into three columns:

- Capture:** Contains the "Interface List" section, which shows a live list of capture interfaces (counting incoming packets). Below this, there are options to "Start capture on interface:" with a list of interfaces: Atheros L1C PCI-E Ethernet Controller, Broadcom, Microsoft, and Microsoft. There is also a "Capture Options" section for starting a capture with detailed options.
- Files:** Contains an "Open" section for opening a previously captured file, an "Open Recent:" section, and "Sample Captures" which are example capture files on the wild.
- Online:** Contains links to the "Website", "User's Guide" (local version if installed), and "Security" (work with Wireshark as securely as possible).

At the bottom, there is a status bar showing "Ready to load or capture" and "No Packets". The taskbar at the very bottom shows several open applications: Internet Explorer, Dropbox, Wireshark, wireshark\_intro.p..., USING WIRESHARK, and The Wireshark Ne... The system tray shows the date and time as 16:45 on 2012/1/25.

# CAPTURING DATA

- Click on the specific interface you want to capture traffic from.

The screenshot shows the Wireshark interface with a packet capture list containing four entries:

No.	Time	Source	Destination	Protocol	Info
1	0.000000	67.228.110.120	192.168.0.100	TCP	http > 1232 [FIN, ACK] Seq=1 Ack=1 win=65 Len=0
2	0.000073	192.168.0.100	67.228.110.120	TCP	1232 > http [ACK] Seq=1 Ack=2 win=4313 Len=0
3	1.990387	192.168.0.100	67.228.110.120	TCP	1232 > http [FIN, ACK] Seq=1 Ack=2 win=4313 Len=0
4	2.019462	67.228.110.120	192.168.0.100	TCP	http > 1232 [ACK] Seq=2 Ack=2 win=65 Len=0

Below the packet list, the details pane shows the structure of the first packet:

- Frame 1 (54 bytes on wire, 54 bytes captured)
- Ethernet II, Src: D-Link\_cf:ea:c7 (00:24:01:cf:ea:c7), Dst: NonhaIPr\_77:5d:a1 (00:25:56:77:5d:a1)
- Internet Protocol, Src: 67.228.110.120 (67.228.110.120), Dst: 192.168.0.100 (192.168.0.100)
- Transmission Control Protocol, Src Port: http (80), Dst Port: 1232 (1232), Seq: 1, Ack: 1, Len: 0

The packet bytes pane shows the raw data in hexadecimal and ASCII:

```
0000 00 25 56 77 5d a1 00 24 01 cf ea c7 08 00 45 00  .5Vw)..$ .....E.  
0010 00 28 8c a0 40 00 36 06 44 c7 43 e4 6e 78 c0 a8  .(.0.0. D.C.mx..  
0020 00 64 00 50 04 d0 42 4e 7c f5 d3 a4 16 85 50 11  .d.P..BN |.....P.  
0030 00 41 8d 9c 00 00                                .A....
```



# ANALYZING CAPTURED DATA

No. .	Time	Source	Destination	Protocol	Info
154	97.802301	192.168.0.100	174.129.27.168	TLSv1	Application Data, Application Data, Application Data,
155	97.805312	192.168.0.100	174.129.27.168	TLSv1	Application Data,
156	97.848793	174.129.27.168	192.168.0.100	TCP	https > bvcontrol [ACK] Seq=1414 Ack=3545 win=16896 Len=0
157	97.848865	192.168.0.100	174.129.27.168	TLSv1	Application Data, Application Data, Application Data,
158	97.848872	192.168.0.100	174.129.27.168	TLSv1	Application Data, Application Data, Application Data,
159	97.890781	174.129.27.168	192.168.0.100	TCP	https > bvcontrol [ACK] Seq=1414 Ack=4993 win=19968 Len=0
160	97.890856	192.168.0.100	174.129.27.168	TCP	[TCP segment of a reassembled PDU]
161	97.890864	192.168.0.100	174.129.27.168	TCP	[TCP segment of a reassembled PDU]
162	97.897797	174.129.27.168	192.168.0.100	TCP	https > bvcontrol [ACK] Seq=1414 Ack=6441 win=23040 Len=0
163	97.897850	192.168.0.100	174.129.27.168	TCP	[TCP segment of a reassembled PDU]

Time of capturing the packet

Source IP

Destination IP

Protocol Name

Brief description of the packet data

# ANALYZING CAPTURED DATA

No. -	Time	Source	Destination	Protocol	Info
154	97.803307	192.168.0.100	174.129.27.168	TLSv1	Application Data, Application
155	97.805312	192.168.0.100	174.129.27.168	TLSv1	Application Data, Application
156	97.848793	174.129.27.168	192.168.0.100	TCP	https > bvcontrol [ACK] Seq=14
157	97.848865	192.168.0.100	174.129.27.168	TLSv1	Application Data, Application
158	97.848872	192.168.0.100	174.129.27.168	TLSv1	Application Data, Application
159	97.890781	174.129.27.168	192.168.0.100	TCP	https > bvcontrol [ACK] Seq=14
160	97.890856	192.168.0.100	174.129.27.168	TCP	[TCP segment of a reassembled
161	97.890864	192.168.0.100	174.129.27.168	TCP	[TCP segment of a reassembled
162	97.897797	174.129.27.168	192.168.0.100	TCP	https > bvcontrol [ACK] Seq=14
163	97.897850	192.168.0.100	174.129.27.168	TCP	[TCP segment of a reassembled

[-] Frame 159 (54 bytes on wire, 54 bytes captured)  
[-] Ethernet II, Src: D-Link\_cf:ea:c7 (00:24:01:cf:ea:c7), Dst: HonHaiPr\_77:5d:a1 (00:25:56:77:5d:a1)  
[-] Internet Protocol, Src: 174.129.27.168 (174.129.27.168), Dst: 192.168.0.100 (192.168.0.100)  
[-] Transmission Control Protocol, Src Port: https (443), Dst Port: bvcontrol (1236), Seq: 1414, Ack: 4993, Len: 0

Hierarchical View

Frame (Bottom Layer)  
Ethernet  
IP  
TCP (Top Layer)

- Note: The hierarchical display here is upside down compared to the Internet protocol stack that you learn in the lecture.

# ANALYZING CAPTURED DATA

The screenshot shows the Wireshark interface with a filter set to 'http.request.version=="HTTP/1.1"'. The packet list pane shows several HTTP GET requests for image files. The selected packet (No. 686) is expanded to show its details, including the Hypertext Transfer Protocol section with the following headers:

```
GET /thumbnail/76d800a7jw1dpgsbd8ja6j.jpg HTTP/1.1\r\nHost: ww1.sinaimg.cn\r\nConnection: keep-alive\r\nUser-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.7 (KHTML, like Gecko) Chrome/16.0.912.77 Safari/535.7\r\nAccept: */*\r\nReferer: http://www.weibo.com/u/1740944337?wvr=3.6&lf=reg\r\nAccept-Encoding: gzip, deflate, sdch\r\nAccept-Language: en-US,en;q=0.8\r\nAccept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3\r\n\r\n[Full request URI: http://ww1.sinaimg.cn/thumbnail/76d800a7jw1dpgsbd8ja6j.jpg]
```

- HTTP header

# WIRESHARK FILTERS

- **Two types of filters:**
  - Capture Filters
  - Display Filters
- Wireshark contains a powerful **capture** filter engine that helps **remove unwanted packets** from a packet trace and only retrieves the packets of our interest.
- **Display** filters let you compare the fields within a protocol against a specific value, compare fields against fields, and check the existence of specified fields or protocols

# EXAMPLE OF A CAPTURE FILTER

Capture

Interface: Local Microsoft: \Device\NPF\_{C372DBF0-E317-4323-96CD-9A93BB8} (selected)

IP address: fe80::b8d8:a9c3:ac04:ce48, 192.168.0.100

Link-layer header type: Ethernet

Capture packets in promiscuous mode

Capture packets in pcap-ng format (experimental)

Limit each packet to 1 bytes

Buffer size: 1 megabyte(s)

Wireless Settings

Remote Settings

Capture Filter: (host 192.168.0.100) || (host 174.36.30.66)

Capture File(s)

File:

Use multiple files

Next file every 1 megabyte(s)

Next file every 1 minute(s)

Ring buffer with 2 files

Stop capture after 1 file(s)

Automatic scrolling in live capture

Hide capture info dialog

Name Resolution

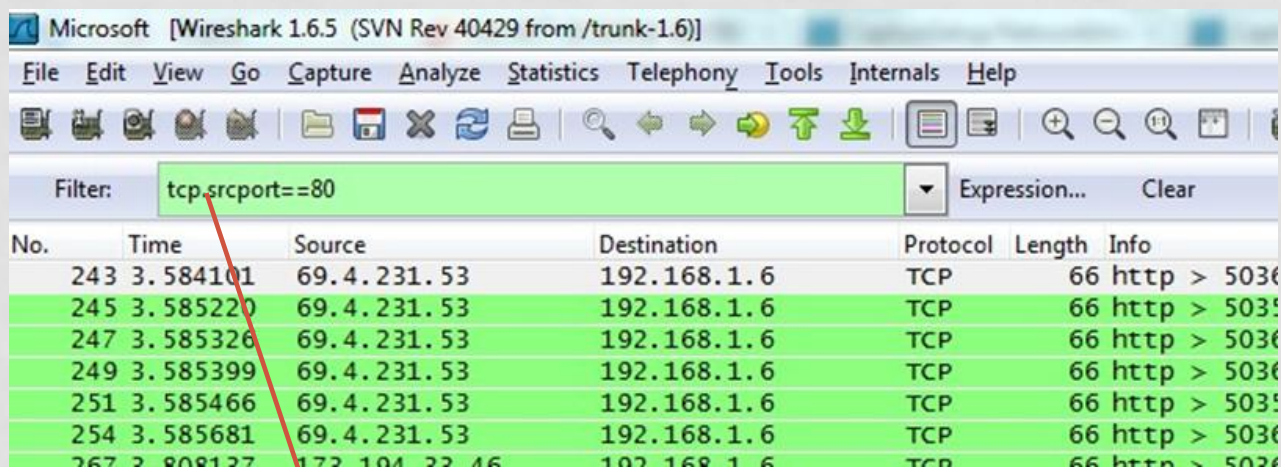
Enable MAC name resolution

Stop Capture ...

... after 1 packet(s)



# EXAMPLE OF A DISPLAY FILTER



Microsoft [Wireshark 1.6.5 (SVN Rev 40429 from /trunk-1.6)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: `tcp.srcport==80` Expression... Clear

No.	Time	Source	Destination	Protocol	Length	Info
243	3.584101	69.4.231.53	192.168.1.6	TCP	66	http > 5036
245	3.585220	69.4.231.53	192.168.1.6	TCP	66	http > 5039
247	3.585326	69.4.231.53	192.168.1.6	TCP	66	http > 5036
249	3.585399	69.4.231.53	192.168.1.6	TCP	66	http > 5036
251	3.585466	69.4.231.53	192.168.1.6	TCP	66	http > 5039
254	3.585681	69.4.231.53	192.168.1.6	TCP	66	http > 5036
267	3.808127	172.16.17.16	192.168.1.6	TCP	66	http > 5036

- Display filter separates the packets to be displayed (In this case, only packets with source port 80 are displayed)

# WIRESHARK FILTERS

- **Comparison operators**
- Fields can also be compared against values. The comparison operators can be expressed either through English-like abbreviations or through C-like symbols:
  - eq, == Equal
  - ne, != Not Equal
  - gt, > Greater Than
  - lt, < Less Than
  - ge, >= Greater than or Equal to
  - le, <= Less than or Equal to

# WIRESHARK FILTERS

- **Logical Expressions**

Tests can be combined using logical expressions. These too are expressible in C-like syntax or with English-like abbreviations:

and, && Logical AND

or, || Logical OR

not, ! Logical NOT

- Some Valid Filters
- `tcp.port == 80 and ip.src == 192.168.2.1`
- `http and frame[100-199] contains "wireshark"`



# WIRESHARK FILTERS

- **The Slice Operator**
- You can take a slice of a field if the field is a text string or a byte array. For example, you can filter the HTTP header fields. Here the header “location” indicates the REDIRECTION happens.

```
http.location[0:4]=="http"
```

- Another example is:

```
http.content_type[0:4] == "text"
```

# CAPTURE FILTERS

Syntax	Protocol	Direction	Host(s)	Logical Op.	Other Express.
Example	tcp	dst	136.159.5.20	and	host 136.159.5.6

- **Protocol:**
  - *Values:* ether, fddi, ip, arp, rarp, decnet, lat, sca, moprc, mopdl, tcp and udp.
  - If no protocol is specified, all the protocols are used.
- **Direction:**
  - *Values:* src, dst, src and dst, src or dst
  - If no source or destination is specified, the "src or dst" keywords are applied.
  - For example, "host 136.159.5.20" is equivalent to "src or dst host 136.159.5.20".

# CAPTURE FILTERS

- **Host(s):**
  - Values: net, port, host, portrange.
  - If no host(s) is specified, the "host" keyword is used.
  - For example, "src 136.159.5.20" is equivalent to "src host 136.159.5.20".
- **Logical Operations:**
  - Values: not, and, or.
  - Negation ("not") has highest precedence. Alternation ("or") and concatenation ("and") have equal precedence and associate left to right.
  - For example,  
"not tcp port 3128 and tcp port 80" is equivalent to "(not tcp port 3128) and tcp port 80".

## CAPTURE FILTERS(EXAMPLES)

- **tcp port 80**  
Displays packets with tcp protocol on port 80.
- **ip src host 136.159.5.20**  
Displays packets with source IP address equals to 136.159.5.20.
- **host 136.159.5.1**  
Displays packets with source or destination IP address equals to 136.159.5.1.
- **src portrange 2000-2500**  
Displays packets with source UDP or TCP ports in the 2000-2500 range.

## CAPTURE FILTERS(EXAMPLES)

- **src host 136.159.5.20 and not dst host 136.159.5.1**

Displays packets with source IP address equals to 136.159.5.20 and in the same time not with the destination IP address 136.159.5.1.

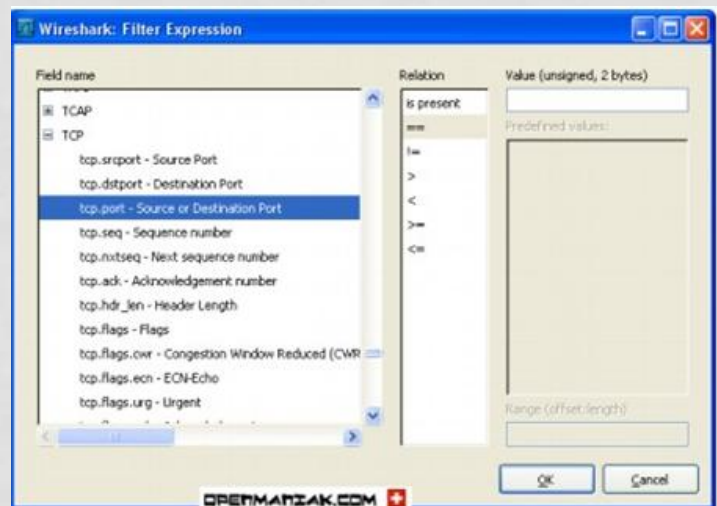
- **(src host 136.159.5.1 or src host 136.159.5.3) and tcp dst portrange 200-10000 and dst host 136.159.5.2**

Displays packets with source IP address 136.159.5.1 or source address 136.159.5.3, the result is then concatenated with packets having destination TCP portrange from 200 to 10000 and destination IP address 136.159.5.2.

# DISPLAY FILTERS

Syntax	Protocol	.	String 1	.	String 2	Comparison operators	Value	Logical Op.	Other Expr.
Example	http	.	request	.	method	==	get	or	tcp.port == 80

- String1 , String2 (Optional settings): Sub protocol categories inside the protocol. To find them, look for a protocol and then click on the "+" character.



## DISPLAY FILTERS(EXAMPLES)

- **ip.addr == 136.159.5.20**  
Displays the packets with source or destination IP address equals to 136.159.5.20 .
- **http.request.version=="HTTP/1.1"**  
Display http Version
- **tcp.dstport == 25**
- **tcp.flags**  
Display packets having a TCP flags
- **tcp.flags.syn == 0x02**  
Display packets with a TCP SYN flag. (Synchronize sequence numbers. Only the first packet sent from each end should have this flag set)