

# CSCI369 Ethical Hacking

## Lecture 2-1: TCP/IP Basics & Capturing Traffic

A/Prof Joonsang Baek

School of Computing and Information Technology



This slide is copyrighted. It must not be distributed without permission from UOW

# Overview of TCP/IP

- The TCP/IP layer (TCP/IP stack)

| OSI model    |   |                          |                         |   |
|--------------|---|--------------------------|-------------------------|---|
| Layer        |   | Protocol data unit (PDU) | Function <sup>[5]</sup> |   |
| Host layers  | 7 | Application              | Data                    | High-level APIs, including resource sharing, remote file access   |
|              | 6 | Presentation             |                         | Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption   |
|              | 5 | Session                  |                         | Managing communication sessions, i.e. continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes |
|              | 4 | Transport                | Segment, Datagram       | Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing                    |
| Media layers | 3 | Network                  | Packet                  | Structuring and managing a multi-node network, including addressing, routing and traffic control  |
|              | 2 | Data link                | Frame                   | Reliable transmission of data frames between two nodes connected by a physical layer  |
|              | 1 | Physical                 | Symbol                  | Transmission and reception of raw bit streams over a physical medium  |

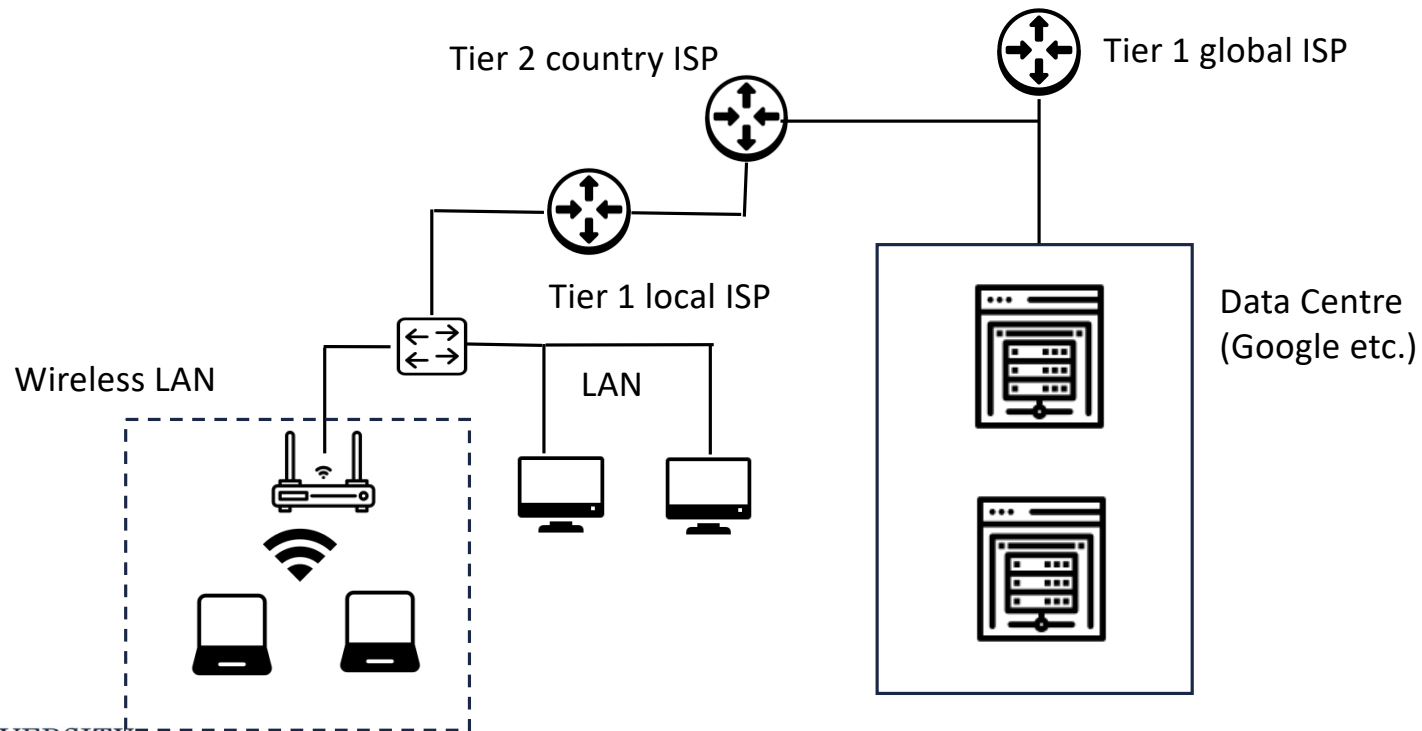
(From [https://en.wikipedia.org/wiki/OSI\\_model](https://en.wikipedia.org/wiki/OSI_model))



UNIVERSITY  
OF WOLLONGONG  
AUSTRALIA

This slide is copyrighted. It must not be distributed without permission from UOW

# Overview of Network Hierarchy



UNIVERSITY  
OF WOLLONGONG  
AUSTRALIA

This slide is copyrighted. It must not be distributed without permission from UOW

# The Application Layer

- Some important applications
  - **http**: The primary protocol used to communicate over the web.
  - **https**: The secure version of http based on TLS (= SSL).
  - **FTP**: Allows different OS's to transfer files between one another.  
(Should not use anonymous username with no password.)
  - **SMTP** (Simple Mail Transfer Protocol) : Transmits email messages across the Internet.

# The Application Layer

- Some important applications
  - **SSH**: Enables users to *securely* log on to a remote server and issue commands interactively.
  - **Telnet**: Enables users to *insecurely* log on to a remote server and issue commands interactively. (**Username and password are not encrypted**. Do not use!)

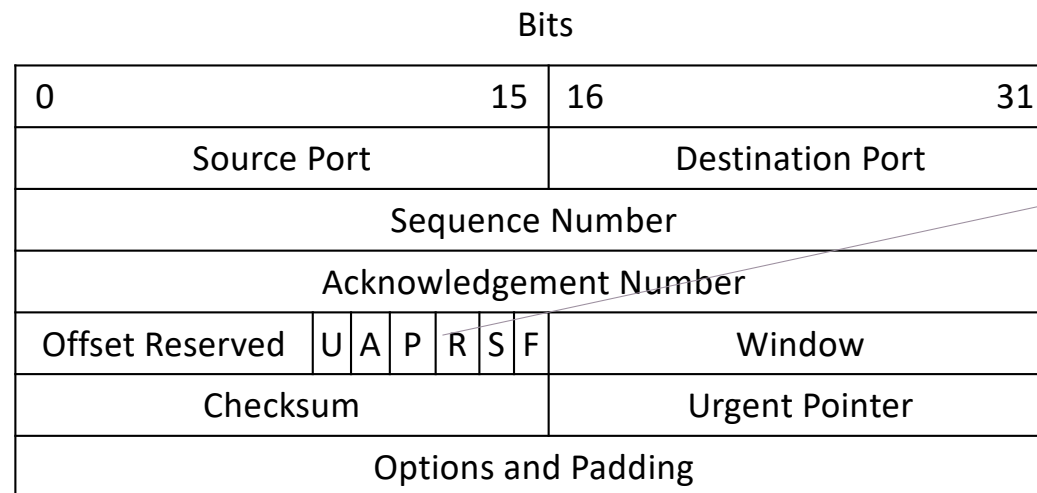
# The Transport Layer

- Main function
  - Data is encapsulated into segments
  - Segments can be TCP or UDP
- TCP
  - **Connection-oriented protocol**: The source (sender) cannot send any data to the destination (receiver) until the destination node acknowledges that it's listening to the source.
    - ✓ The source sends a **SYN** packet to the destination
    - ✓ The destination sends **SYN-ACK** to the source
    - ✓ The source sends **ACK** to the destination

Three-way  
handshake

# The Transport Layer

- TCP Segment Header



TCP Flags

TCP Header Diagram

# The Transport Layer

- TCP Flags

- Can be set 0 (off) or 1 (on)

- ✓ SYN flag – This signifies the beginning of a session
    - ✓ ACK flag – This acknowledges a connection request
    - ✓ PSH flag – This flag is used to deliver data directly to an application (Data is not buffered; it is sent immediately)
    - ✓ URG flag – This flag is used to signify urgent data
    - ✓ RST flag – This resets or drops a connection
    - ✓ FIN flag – This indicates that the connection is finished

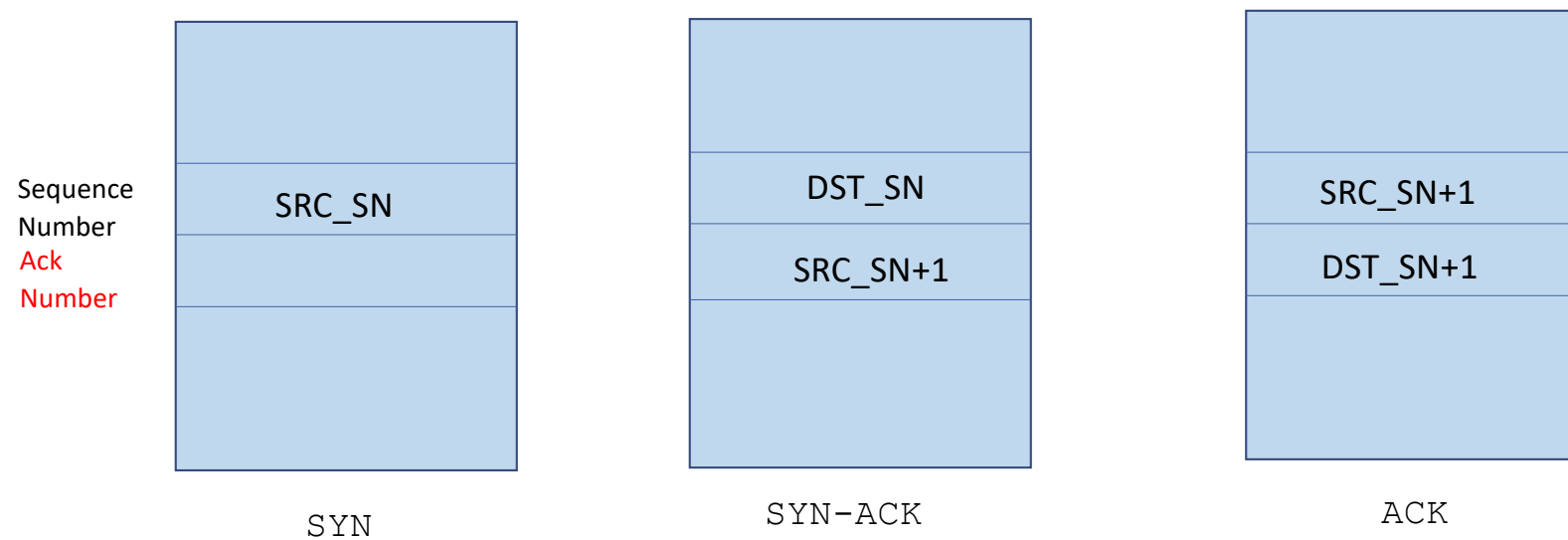


# The Transport Layer

- Three-way handshake
  1. A sequence number from the source node (SRC\_SN) is put in the Sequence Number field (32 bits) of a SYN segment and is sent to the destination node.
  2. The destination node extracts the SRC\_SN from the SYN segment and puts SRC\_SN+1 in the Acknowledgement Number field; also the destination node selects a DST\_SN and puts it in the Sequence Number field of a SYN-ACK segment, which is sent to the source node.
  3. The source node extracts the DST\_SN from the SYN-ACK segment and puts DST\_SN+1 in the Acknowledgement Number field; also the destination node puts SRC\_SN+1 in the sequence number field of a ACK segment, which is sent to the destination node.

# The Transport Layer

- Illustration - Three-way handshake with SN



# The Transport Layer

- TCP Ports

- 16 bits for both source and destination ports
- A port is a logical component of TCP connection assigned to a process requiring network connectivity.
- A port is the way a client program specifies a specific server program.
- **Some important ports (to remember)**
  - ✓ 21: FTP
  - ✓ 22: SSH
  - ✓ 25: SMTP
  - ✓ 53: DNS
  - ✓ 80: HTTP
  - ✓ 443: HTTPS (Secure HTTP)
  - ✓ 110: POP3 (Post Office Protocol 3)

# The Transport Layer

- UDP

- UDP does not need to verify whether the destination is listening or ready to accept the packets

- ✓ It has no handshaking dialogues

- ✓ No guarantee of delivery, ordering and/or duplicate protection

- **Connectionless** transmission

- Unreliable but fast

- ✓ Used in applications in which queries must be fast and only consist of a single request such as **DNS and DHCP**

## Frequently-Used Port Numbers

| PROTOCOL NAME     | PORT #                                    | PROTOCOL NAME | PORT #               |
|-------------------|---|---------------|----------------------|
| FTP               | TCP 21                                    | LDAP over SSL | TCP 636              |
| SSH/SCP           | TCP 22                                    | FTP over SSL  | TCP 989–990          |
| Telnet            | TCP 23                                    | IMAP over SSL | TCP 993              |
| SMTP              | TCP 25                                    | POP3 over SSL | TCP 995              |
| DNS Query         | UDP 53                                    | MS-SQL        | TCP 1433             |
| DNS Zone Transfer | TCP 53                                    | NFS           | TCP 2049             |
| DHCP              | UDP 67<br>UDP 68                          | Docker Daemon | TCP 2375             |
| TFTP              | UDP 69                                    | Oracle DB     | TCP 2483–2484        |
| HTTP              | TCP 80                                    | MySQL         | TCP 3306             |
| Kerberos          | UDP 88                                    | RDP           | TCP 3389             |
| POP3              | TCP 110                                   | VNC           | TCP 5500             |
| SNMP              | UDP 161<br>UDP 162                        | PCAnywhere    | TCP 5631             |
| NetBIOS           | TCP/UDP 137<br>TCP/UDP 138<br>TCP/UDP 139 | IRC           | TCP 6665–6669        |
| IMAP              | TCP 143                                   | IRC SSL       | TCP 6679<br>TCP 6697 |
| LDAP              | TCP 389                                   | BitTorrent    | TCP 6881–6999        |
| HTTPS (TLS)       | TCP 443                                   | Printers      | TCP 9100             |
| SMTP over SSL     | TCP 465                                   | WebDAV        | TCP 9800             |
| rlogin            | TCP 513                                   | Webmin        | 10000                |



UNIVERSITY  
OF WOLLONGONG  
AUSTRALIA

This slide  
permission from CC BY

# The Internet Layer

- Main functionality
  - Responsible for **routing a packet (datagram) to a destination address**
  - Routing is done by using an IP address
  - Connectionless transmission like UDP

# IP Address Basics (IPv4)

- IP address: Four integers separated by decimals; each integer represents 1 byte = 8 bits = 1 *octet*
  - Ex) 128.214.18.16 = 1000000.11010110.00010010.00010000 (Binary)
- Each IP address is separated into a network address and a host address → One part of the IP address represents a network prefix and the other part represents a host
  - ✓ Ex) In the classful IP addressing, 192.168.1.231 belongs to Class C address, which uses the first three numbers to identify the network and the last number to identify the host

# Subnet mask

- A bitmask that yields the network prefix (network address) when applied by a **bitwise AND operation with any IP address** in the network
- Example
  - Network prefix of an IP address 192.168.54.3 with a subnet mask 255.255.255.0 is 192.168.54
  - An IP address 192.168.54.3 with a subnet mask 255.255.0.0 produces a network prefix 192.168 → This means that to be on the same network, two machines must have IP addresses starting with 192.168
  - Hence,
    - ✓ Subnet mask for Class A: 255.0.0.0
    - ✓ Subnet mask for Class B: 255.255.0.0
    - ✓ Subnet mask for Class C: 255.255.255.0



# CIDR IP Addressing

- **CIDR (Classless Inter-Domain Routing)** addressing is a compact method for representing an IP address and its associated network prefix.
- The CIDR notation is constructed from an IP address, a slash ('/') character, and a decimal number. The trailing number is the count of leading 1 bits in the **subnet mask**.
  - Form: a.b.c.d/x
- Motivation: The classful network does not fully represent more fine-grained network prefixes. CIDR resolves this issue.

# CIDR IP Addressing

Note that /8, /16, /24 represent Class A, Class B and Class C, respectively.



UNIVERSITY  
OF WOLLONGONG  
AUSTRALIA

This slide is copyright  
permission from UC

| CIDR | Subnet mask<br>(decimal) | Subnet mask<br>(binary)             | Available addresses |                 |
|------|--------------------------|-------------------------------------|---------------------|-----------------|
| /0   | 0.0.0.0                  | 00000000.00000000.00000000.00000000 | 4,294,967,296       | 2 <sup>32</sup> |
| /1   | 128.0.0.0                | 10000000.00000000.00000000.00000000 | 2,147,483,648       | 2 <sup>31</sup> |
| /2   | 192.0.0.0                | 11000000.00000000.00000000.00000000 | 1,073,741,824       | 2 <sup>30</sup> |
| /3   | 224.0.0.0                | 11100000.00000000.00000000.00000000 | 536,870,912         | 2 <sup>29</sup> |
| /4   | 240.0.0.0                | 11110000.00000000.00000000.00000000 | 268,435,456         | 2 <sup>28</sup> |
| /5   | 248.0.0.0                | 11111000.00000000.00000000.00000000 | 134,217,728         | 2 <sup>27</sup> |
| /6   | 252.0.0.0                | 11111100.00000000.00000000.00000000 | 67,108,864          | 2 <sup>26</sup> |
| /7   | 254.0.0.0                | 11111110.00000000.00000000.00000000 | 33,554,432          | 2 <sup>25</sup> |
| /8   | 255.0.0.0                | 11111111.00000000.00000000.00000000 | 16,777,216          | 2 <sup>24</sup> |
| /9   | 255.128.0.0              | 11111111.10000000.00000000.00000000 | 8,388,608           | 2 <sup>23</sup> |
| /10  | 255.192.0.0              | 11111111.11000000.00000000.00000000 | 4,194,304           | 2 <sup>22</sup> |
| /11  | 255.224.0.0              | 11111111.11100000.00000000.00000000 | 2,097,152           | 2 <sup>21</sup> |
| /12  | 255.240.0.0              | 11111111.11110000.00000000.00000000 | 1,048,576           | 2 <sup>20</sup> |
| /13  | 255.248.0.0              | 11111111.11111000.00000000.00000000 | 524,288             | 2 <sup>19</sup> |
| /14  | 255.252.0.0              | 11111111.11111100.00000000.00000000 | 262,144             | 2 <sup>18</sup> |
| /15  | 255.254.0.0              | 11111111.11111110.00000000.00000000 | 131,072             | 2 <sup>17</sup> |
| /16  | 255.255.0.0              | 11111111.11111111.00000000.00000000 | 65,536              | 2 <sup>16</sup> |
| /17  | 255.255.128.0            | 11111111.11111111.10000000.00000000 | 32,768              | 2 <sup>15</sup> |
| /18  | 255.255.192.0            | 11111111.11111111.11000000.00000000 | 16,384              | 2 <sup>14</sup> |
| /19  | 255.255.224.0            | 11111111.11111111.11100000.00000000 | 8,192               | 2 <sup>13</sup> |
| /20  | 255.255.240.0            | 11111111.11111111.11110000.00000000 | 4,096               | 2 <sup>12</sup> |
| /21  | 255.255.248.0            | 11111111.11111111.11111000.00000000 | 2,048               | 2 <sup>11</sup> |
| /22  | 255.255.252.0            | 11111111.11111111.11111100.00000000 | 1,024               | 2 <sup>10</sup> |
| /23  | 255.255.254.0            | 11111111.11111111.11111110.00000000 | 512                 | 2 <sup>9</sup>  |
| /24  | 255.255.255.0            | 11111111.11111111.11111111.00000000 | 256                 | 2 <sup>8</sup>  |
| /25  | 255.255.255.128          | 11111111.11111111.11111111.10000000 | 128                 | 2 <sup>7</sup>  |
| /26  | 255.255.255.192          | 11111111.11111111.11111111.11000000 | 64                  | 2 <sup>6</sup>  |
| /27  | 255.255.255.224          | 11111111.11111111.11111111.11100000 | 32                  | 2 <sup>5</sup>  |
| /28  | 255.255.255.240          | 11111111.11111111.11111111.11110000 | 16                  | 2 <sup>4</sup>  |
| /29  | 255.255.255.248          | 11111111.11111111.11111111.11111000 | 8                   | 2 <sup>3</sup>  |
| /30  | 255.255.255.252          | 11111111.11111111.11111111.11111100 | 4                   | 2 <sup>2</sup>  |
| /31  | 255.255.255.254          | 11111111.11111111.11111111.11111110 | 2                   | 2 <sup>1</sup>  |
| /32  | 255.255.255.255          | 11111111.11111111.11111111.11111111 | 1                   | 2 <sup>0</sup>  |

# CIDR IP Addressing

- Example 1) What IP ranges does 192.168.101.3/22 represent?
  - First, calculate the **network prefix** by applying "AND ( $\wedge$ )" to bit representations of IP address (192.168.101.3) and subnet mask (/22).

|                |                 |                 |                 |                 |
|----------------|-----------------|-----------------|-----------------|-----------------|
| IP address     | 192             | 168             | 101             | 3               |
|                | 11000000        | 10101000        | 01100101        | 00000011        |
| /22            | 11111111        | 11111111        | 11111100        | 00000000        |
| Network Prefix | <b>11000000</b> | <b>10101000</b> | <b>01100100</b> | <b>00000000</b> |
|                | 192             | 168             | 100             | 0               |

NB: Binary calculator <https://www.calculator.net/binary-calculator.html>

# CIDR IP Addressing

- Then, calculate how many addresses are available:

It is  $2^{32-22}=2^{10}=1024$ .

✓ Note that  $1024/256=4$ .

- Therefore, it represents IPs from 192.168.100.0 to 192.168.100.255, from 192.168.101.0 to 192.168.101.255, from 192.168.102.0 to 192.168.102.255 and from 192.168.103.0 to 192.168.103.255.
- Therefore, the range is 192.168.100.0 to 192.168.103.255.

# CIDR IP Addressing

- Example 2) Do CIDR notations 10.10.1.45/27 and 10.10.1.61/27 have the same network prefix? How about 10.10.1.65/27?

➤ First, calculate the network prefix of 10.0.1.45/27.

|                |                 |                 |                 |                 |
|----------------|-----------------|-----------------|-----------------|-----------------|
| IP address     | 10              | 10              | 1               | 45              |
|                | 00001010        | 00001010        | 00000001        | 00101101        |
| /27            | 11111111        | 11111111        | 11111111        | 11100000        |
| Network Prefix | <b>00001010</b> | <b>00001010</b> | <b>00000001</b> | <b>00100000</b> |
|                | 10              | 10              | 1               | 32              |

# CIDR IP Addressing

➤ Then, calculate the network prefix for 10.10.1.61.

|                |                 |                 |                 |                 |
|----------------|-----------------|-----------------|-----------------|-----------------|
| IP address     | 10              | 10              | 1               | 61              |
|                | 00001010        | 00001010        | 00000001        | 00111101        |
| /27            | 11111111        | 11111111        | 11111111        | 11100000        |
| Network Prefix | <b>00001010</b> | <b>00001010</b> | <b>00000001</b> | <b>00100000</b> |
|                | 10              | 10              | 1               | 32              |

➤ This implies that hosts ending with .45 and .61 with belong to the same network.

# CIDR IP Addressing

➤ Now, find out the network prefix for 10.10.1.65.

|                |                 |                 |                 |                 |
|----------------|-----------------|-----------------|-----------------|-----------------|
| IP address     | 10              | 10              | 1               | 65              |
|                | 00001010        | 00001010        | 00000001        | 01000001        |
| /27            | 11111111        | 11111111        | 11111111        | 11100000        |
| Network Prefix | <b>00001010</b> | <b>00001010</b> | <b>00000001</b> | <b>01000000</b> |
|                | 10              | 10              | 1               | 64              |

➤ Hence, 10.10.1.65/27 does not belong to the same network as the two previous ones.

# The Number of Host IPs

- We do not use IP address that represents network (network prefix) and the last address in the IP range for **a host IP address**.
  - The last address is reserved for broadcast address.
- For example, in Example 1) **192.168.100.0** is not used and **192.168.103.255** is reserved as a broadcast address.
- Therefore, the number of **actual IP addresses available for hosts** in Example 1) is  $2^{10}-2=1022$ .



# IPv6 Addressing

- Developed to increase the space of IP address (The size of IPv4 address space =  $2^{32}$ .)
- IPv6 uses 16 bytes address: The size of IPv6 address space =  $2^{128}$
- Example:

✓ 1111:0cb7:75a2:0110:1234:3a2e:1113:7777

1111 → 0001 0001 0001 0001 (2 bytes)

1113 → 0001 0001 0001 0011

0cb7 → 0000 1100 1011 0111

7777 → 0111 0111 0111 0111

75a2 → 0111 0101 1010 0010

0110 → 0000 0001 0001 0000

1234 → 0001 0010 0011 0100

3a2e → 0011 1010 0010 1110

128-bit representation of IPv6  
address



# ICMP (Internet Control Message Protocol)

- It is used by network devices, including routers, **to send error messages and operational information** such as “Requested service is not available” or “Destination network unreachable”
- It is used in the ping tool
  - A source sends ICMP ECHO\_REQUEST to a destination system
  - If the system is live, it will respond by sending ICMP ECHO\_REPLY
- It is also used in the traceroute tool
  - ✓ In UNIX-like OS, traceroute has an option (-I) to use ICMP (By default, UDP is used.)
  - ✓ In Windows OS, tracert uses ICMP by default. (No need to use the -I option.)

# Introduction to Wireshark

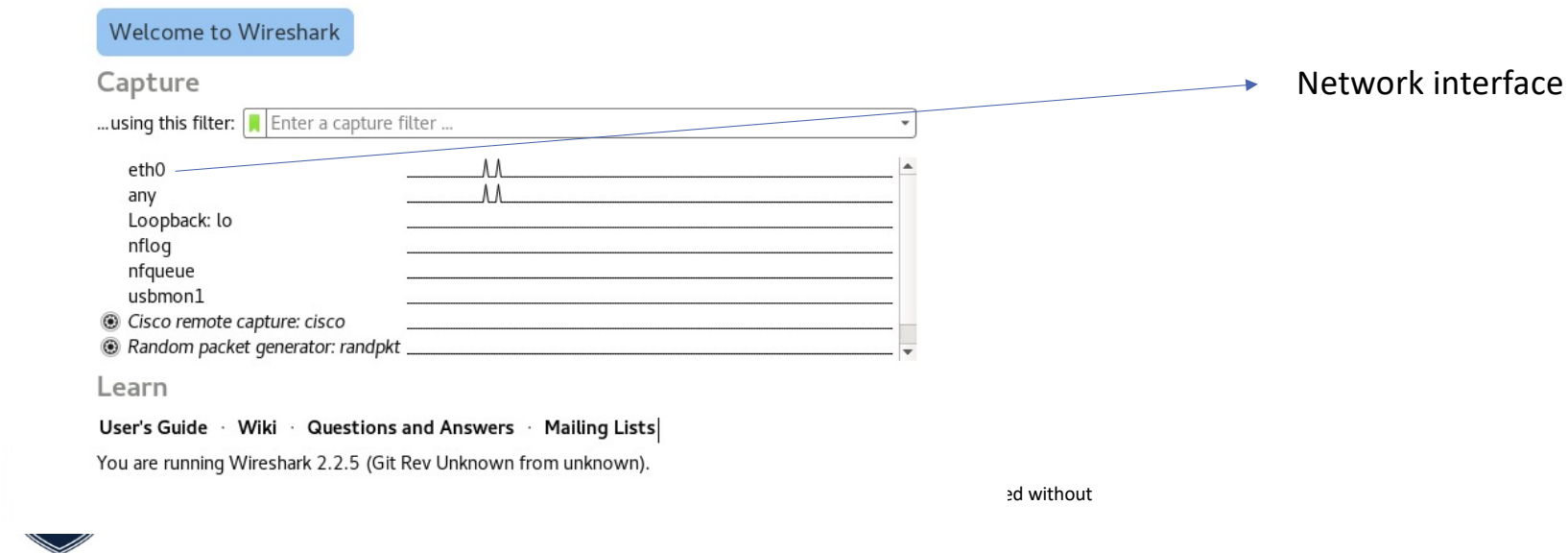
- Wireshark

- A well-known network analysis tool previously known as Ethereal.
- It captures packets in real time and displays them in human-readable format.
- Main features include filters and color coding to analyse network traffic and inspect individual packets.

# Capturing Traffic

- Selecting interfaces

- After launching Wireshark, a user can select a network interface and start capturing packets on that interface.

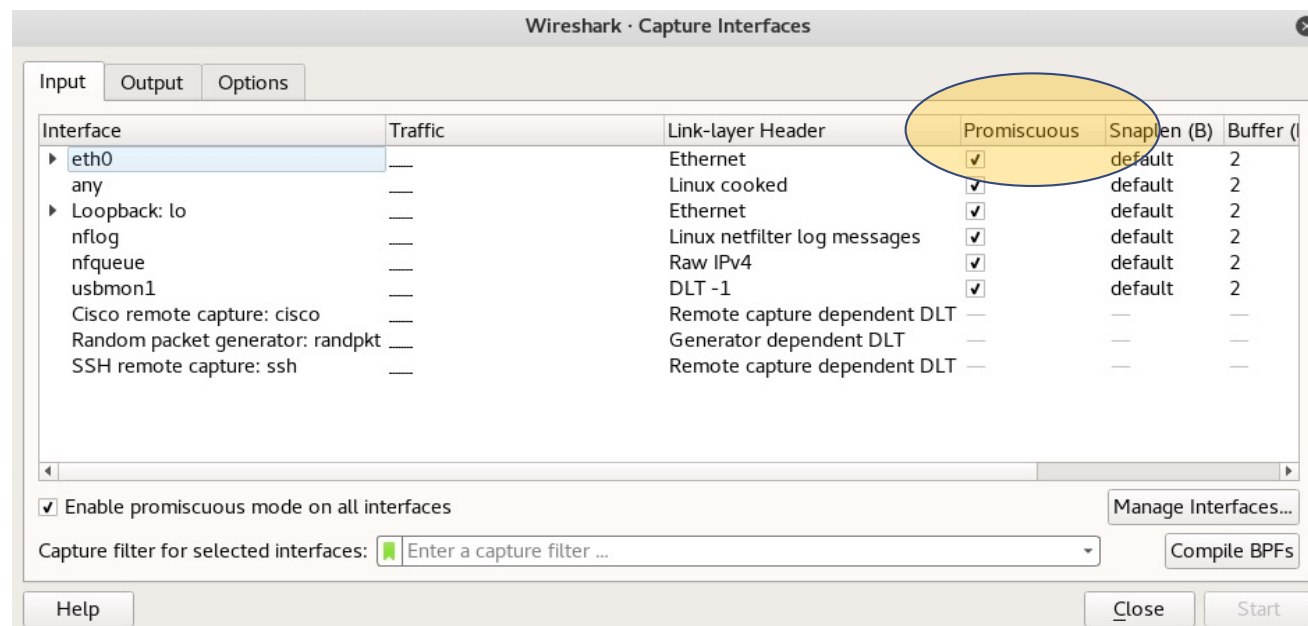


# Capturing Traffic

- **Promiscuous mode**

- Promiscuous mode is a mode for a wired network interface controller (NIC) or wireless network interface controller (WNIC) that causes the controller to **pass all traffic** it receives to the CPU.
- This mode is normally used for packet sniffing which may take place on a router or on a computer connected to a hub or a part of a WLAN.
- By default, Wireshark runs in promiscuous mode but can be updated in the Capture → Option panel

# Capturing Traffic



# Capturing Traffic

- Packet capturing: Upon selecting network interface, the packets start to appear in real time; Wireshark captures each packet sent to or from the system.
- In promiscuous mode, a user can see all the other packets on the network instead of only packets addressed to the user's network adapter.

# Capturing Traffic

Capturing from eth0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/> Expression... +

| No. | Time        | Source            | Destination       | Protocol | Length | Info           |
|-----|-------------|-------------------|-------------------|----------|--------|----------------|
| 1   | 0.000000000 | 10.0.2.5          | 130.130.213.213   | DNS      | 72     | Standard query |
| 2   | 0.000088313 | 10.0.2.5          | 130.130.213.213   | DNS      | 72     | Standard query |
| 3   | 0.000172260 | 10.0.2.5          | 130.130.213.213   | DNS      | 74     | Standard query |
| 4   | 0.000202359 | 10.0.2.5          | 130.130.213.213   | DNS      | 74     | Standard query |
| 5   | 0.000255796 | 10.0.2.5          | 130.130.213.213   | DNS      | 86     | Standard query |
| 6   | 0.000282626 | 10.0.2.5          | 130.130.213.213   | DNS      | 86     | Standard query |
| 7   | 0.002217285 | RealtekU_12:35:00 | Broadcast         | ARP      | 60     | Who has 10.0.  |
| 8   | 0.002222257 | PcsCompu_20:fc:dd | RealtekU_12:35:00 | ARP      | 42     | 10.0.2.5 is at |

Frame 1: 72 bytes on wire (576 bits), 72 bytes captured (576 bits) on interface 0  
 Ethernet II, Src: PcsCompu\_20:fc:dd (08:00:27:20:fc:dd), Dst: RealtekU\_12:35:00 (52:54:00:12:35:00)  
 Internet Protocol Version 4, Src: 10.0.2.5, Dst: 130.130.213.213  
 User Datagram Protocol, Src Port: 56311, Dst Port: 53  
 Domain Name System (query)

```

0000  52 54 00 12 35 00 08 00 27 20 fc dd 08 00 45 00  RT..5... '....E.
0010  00 3a 2b 1d 40 00 40 11 ab 39 0a 00 02 05 82 82  .:+.@. .9.....
0020  d5 d5 db f7 00 35 00 26 64 94 63 2d 01 00 00 01  .....5.& d.c....
0030  00 00 00 00 00 00 03 77 77 77 04 6b 61 6c 69 03  ....w ww.kali.
0040  6f 72 67 00 00 01 00 01                          org.....
  
```

eth0: live capture in progress permission from UOW Packets: 2167 · Displayed: 2167 (100.0%) Profile: Default

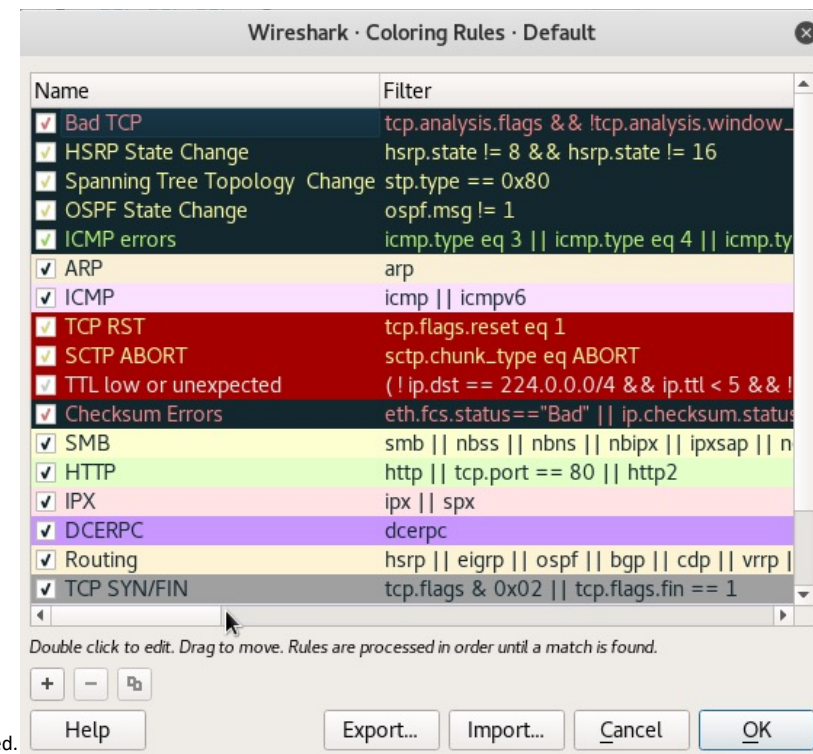


# Color Coding

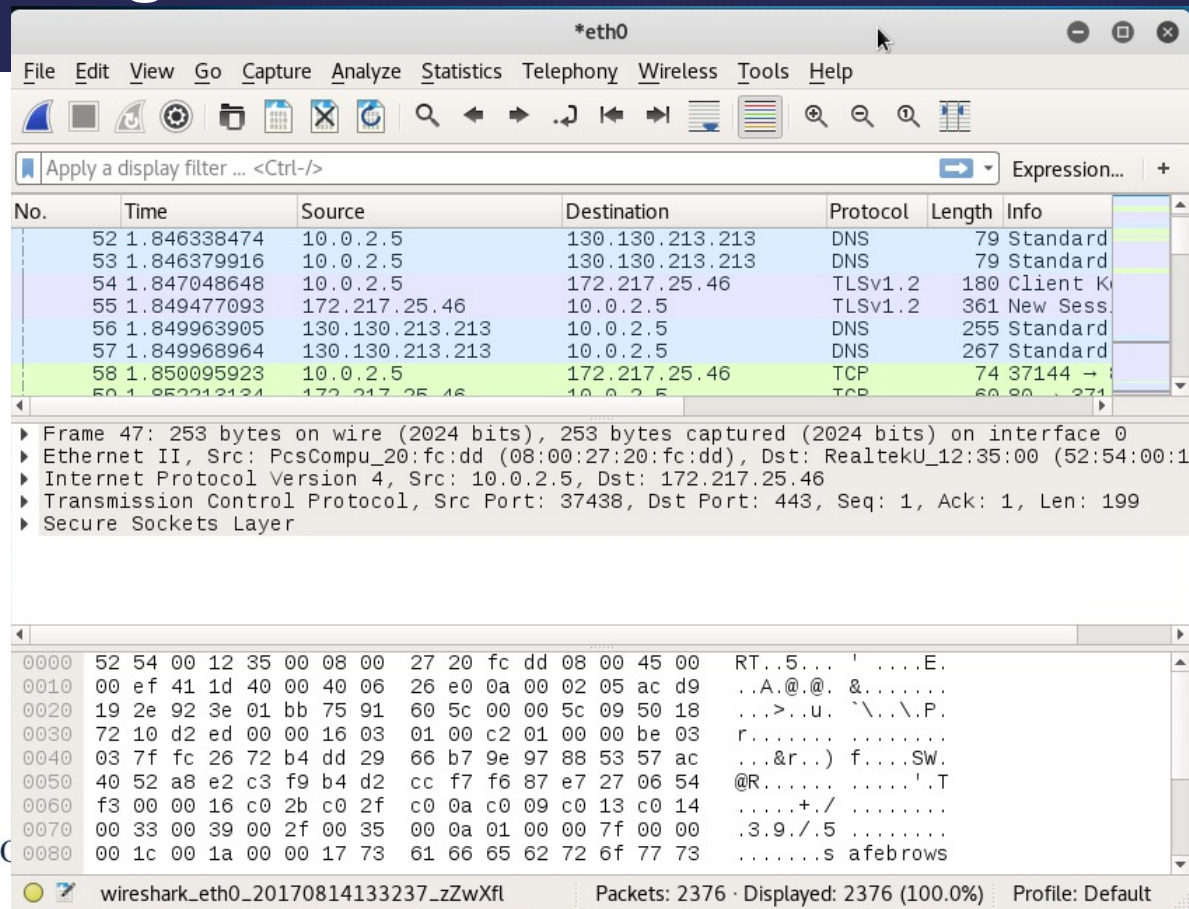
- Wireshark uses colors to identify the types of traffic at a glance.
- Examples (by default):
  - Light purple: TCP traffic
  - Light blue: UDP traffic
  - Black: Packets with errors
- Current (default) color setting can be seen on View → Coloring Rules (Modification is possible)

# Color Coding

- Default color coding rules



# Color Coding



The image shows a Wireshark packet capture window titled '\*eth0'. The interface includes a menu bar (File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help), a toolbar with various icons, and a display filter bar. The main packet list table is color-coded by protocol: DNS (light blue), TLSv1.2 (light purple), and TCP (light green). The details pane for the selected packet (No. 58) shows the following structure:

- Frame 47: 253 bytes on wire (2024 bits), 253 bytes captured (2024 bits) on interface 0
- Ethernet II, Src: PcsCompu\_20:fc:dd (08:00:27:20:fc:dd), Dst: RealtekU\_12:35:00 (52:54:00:12:35:00)
- Internet Protocol Version 4, Src: 10.0.2.5, Dst: 172.217.25.46
- Transmission Control Protocol, Src Port: 37438, Dst Port: 443, Seq: 1, Ack: 1, Len: 199
- Secure Sockets Layer

The packet bytes pane at the bottom shows the raw data in hexadecimal and ASCII format.

Packets captures  
by Wireshark

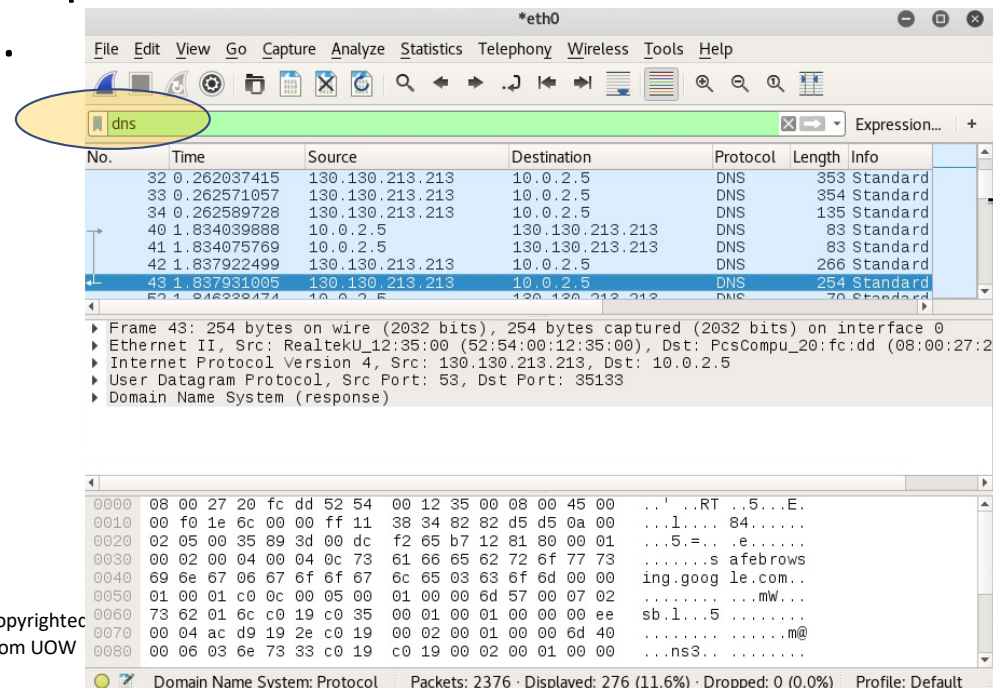


UNIVERSITY  
OF WOLLONGONG  
AUSTRALIA

# Filtering Packets

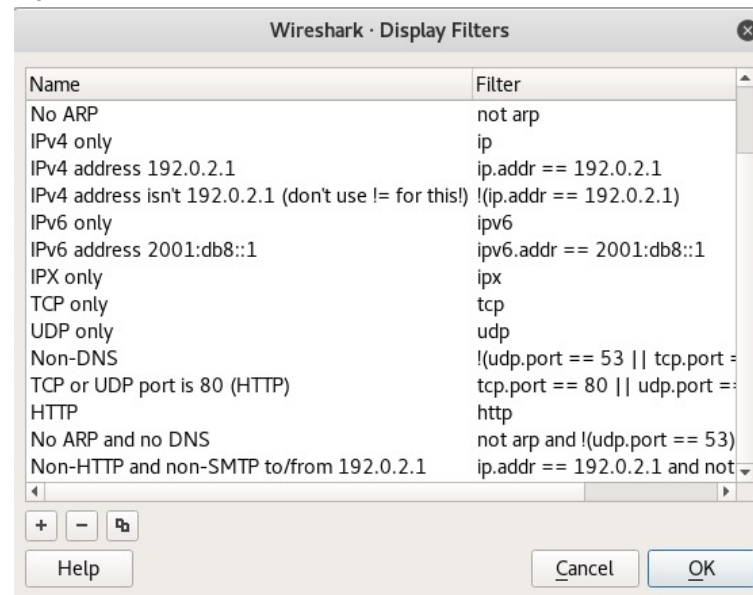
- Filter Box

- The most basic way to filter packets in Wireshark is to enter keywords in the Filter Box.



# Filtering Packets

- Default filters
  - Analyze → Display Filters will list default filters included in Wireshark.



This slide is copyrighted. It must not be distributed without permission from UOW

# Inspecting Packets Example (1)

- Another interesting thing you can do is right-click a packet and select **Follow → TCP Stream**.
  - You'll see the full TCP conversation between the client and the server. You can also click other protocols in the Follow menu to see the full conversations for other protocols, if applicable.
  - Next slide (Screen shot)

# Inspecting Packets Examples (1)

The image displays two overlapping Wireshark windows. The background window, titled 'Wireshark · Follow TCP Stream (tcp.stream eq 2) · wireshark\_eth0\_2017...', shows the details of an HTTP transaction. The top section is an HTTP POST request to '/ocsp' from 'clients1.google.com', using Mozilla/5.0 as the user agent. The bottom section shows the corresponding HTTP 200 OK response, which is an OCSP response. The status bar at the bottom indicates '6 client pkts, 6 server pkts, 9 turns'.

The foreground window, titled 'eth0', shows a packet list with a yellow circle highlighting a specific entry. The packet list shows a SYN packet from 37144 to 80. The packet details pane on the right shows the 'Info' tab for this packet, displaying the sequence number, window size, and length.

POST /ocsp HTTP/1.1  
Host: clients1.google.com  
User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:45.0) Gecko/20100101 Firefox/45.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Content-Length: 75  
Content-Type: application/ocsp-request  
Connection: keep-alive

0IOG0E0C0A0 ..+.....j.....p.I.#z...  
(~d..J.....h.v....b..Z../..PN.....1HTTP/1.1 200 OK  
Content-Type: application/ocsp-response  
Date: Mon, 14 Aug 2017 04:15:09 GMT  
Expires: Fri, 18 Aug 2017 04:15:09 GMT  
Cache-Control: public, max-age=345600  
Server: ocsp\_responder  
Content-Length: 463  
X-XSS-Protection: 1; mode=block  
X-Frame-Options: SAMEORIGIN

0...  
.....0.....+.....  
0.....0.....J.....h.v....b..Z../..  
20170813192729Z0k0i0A0 ..+.....j.....p.I.#z...

6 client pkts, 6 server pkts, 9 turns.

Entire conversation (4700 by ▾ Show and save data as ASCII ▾ Stream 2 ▾  
Find:  Find Next

Info  
37144 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS  
80 → 37144 [SYN, ACK] Seq=0 Ack=1 Win=3276  
37144 → 80 [ACK] Seq=1 Ack=1 Win=29200 Len  
Request  
80 → 37144 [ACK] Seq=1 Ack=430 Win=32339 L  
Response  
37144 → 80 [ACK] Seq=430 Ack=747 Win=30586  
Request

bytes captured (592 bits) on interface 0  
27:20:fc:dd), Dst: RealtekU\_12:35:00 (52:54:  
Dst: 172.217.25.46  
144, Dst Port: 80, Seq: 0, Len: 0

3 00 45 00 RT..5... ' ....E.  
2 05 ac d9 .<..@.@. ....  
0 00 a0 02 .....P4. ....  
3 0a 00 24 r.....\$  
!C.....





## Inspecting Packets Example (2)

- In fact, Wireshark will enable us to capture a username and a password entered to *insecure* website
- A filter you need in this regard is
  - `http.request.method == "POST"`
  - Once you locate a packet right-click the packet and select Follow  
→ TCP Stream
  - Example will follow in the next slide



# Inspecting Packets Example (2)

Wireshark · Follow TCP Stream (tcp.stream eq 7) · wireshark\_en0\_20170814160432\_zQbok8

Wi-Fi: en0

| Destination   | Protocol | Length | Info                           |
|---------------|----------|--------|--------------------------------|
| 45.40.164.136 | HTTP     | 598    | GET / HTTP/1.1                 |
| 10.8.184.133  | HTTP     | 1465   | HTTP/1.1 200 OK (text/html)    |
| 45.40.164.136 | TCP      | 66     | 58170 → 80 [ACK] Seq=533 Ac... |
| 45.40.164.136 | HTTP     | 763    | POST /index.asp HTTP/1.1 (...) |
| 10.8.184.133  | HTTP     | 104    | [TCP Previous segment not c... |
| 10.8.184.133  | TCP      | 1514   | [TCP Out-of-Order] 80 → 581... |
| 45.40.164.136 | TCP      | 78     | [TCP Dup ACK 58#1] 58170 → ... |
| 45.40.164.136 | TCP      | 66     | 58170 → 80 [ACK] Seq=1230 A... |

es captured (6104 bits) on interface 0  
52), Dst: Cisco\_74:f4:00 (00:21:d7:74:f4:00)  
st: 45.40.164.136  
st Port: 80, Seq: 533, Ack: 1400, Len: 697

encoded

00 .!.t..<. ...R..E.  
28 ...@.rq...(  
18 ...:P-H ..)...  
b4 ...;... ..P. ...  
73 ..POST / index.as  
74 p HTTP/1 ..Host  
0a : aavtra in.com..  
70 Connecti on: keep  
2d -alive.. Content-  
68 Length: 63..Cach  
61 e-Contro l: max-a  
74 ge=0..Or igin: ht  
6f tp://aav train.co  
63 m..Uoara de-Insec

2 client pkts, 3 server pkts, 3 turns.

permission from UOW

UNIVE OF WO AUSTRALIA

# Inspecting Packets Example (3)

- By using Wireshark, connections based on the secure application protocol like SSL can be analysed.

|      |           |                |                |         |  |
|------|-----------|----------------|----------------|---------|--|
| 3141 | 36.398756 | 10.9.26.59     | 162.125.34.129 | TLSv1.2 | 237 Client Hello   |
| 3186 | 36.556681 | 162.125.34.129 | 10.9.26.59     | TLSv1.2 | 1514 Server Hello  |
| 3187 | 36.556682 | 162.125.34.129 | 10.9.26.59     | TLSv1.2 | 1514 Certificate [TCP segment of a reassembled PDU]                      |
| 3188 | 36.556684 | 162.125.34.129 | 10.9.26.59     | TLSv1.2 | 156 Server Key Exchange, Server Hello Done                               |
| 3190 | 36.561857 | 10.9.26.59     | 162.125.34.129 | TLSv1.2 | 180 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message |
| 3238 | 36.716422 | 162.125.34.129 | 10.9.26.59     | TLSv1.2 | 296 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message  |
| 3239 | 36.719132 | 10.9.26.59     | 162.125.34.129 | TLSv1.2 | 473 Application Data   |

An example of connection based on SSL