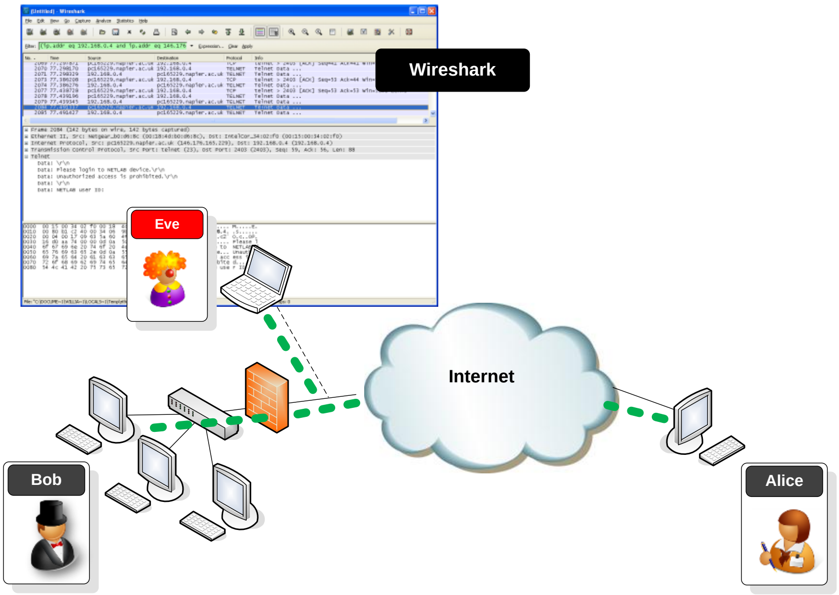
Packet Capture & Traffic Analysis

with Wireshark

Packet Capture (Packet Sniffing)

A **packet sniffer** is an application which can capture and analyse network traffic which is passing through a system's Network Interface Card (NIC). The sniffer sets the card to **promiscuous mode** which means all traffic is read, whether it is addressed to that machine or not. The figure below shows an attacker sniffing packets from the network, and the **Wireshark** packet sniffer/analyser (formerly known as ethereal).



Packet Analysis

Wireshark is an open source cross-platform packet capture and analysis tool, with versions for Windows and Linux. The GUI window gives a detailed breakdown of the network protocol stack for each packet, colorising packet details based on protocol, as well as having functionality to filter and search the traffic, and pick out TCP streams. Wireshark can also save packet data to files for offline analysis and export/import packet captures to/from other tools. Statistics can also be generated for packet capture files.

Wireshark can be used for **network troubleshooting**, to **investigate security issues**, and to **analyse and understand network protocols**. The packet sniffer can exploit information passed in plaintext, i.e. not encrypted. Examples of **protocols** which pass information in plaintext are **Telnet, FTP, SNMP, POP, and HTTP**.

Wireshark is a GUI based network capture tool. There is a command line based version of the packet capture utility, called **TShark**. TShark provides many of the same features as it's big brother, but is console-based. It can be a good alternative if only command line access is available, and also uses less resources as it has no GUI to generate.

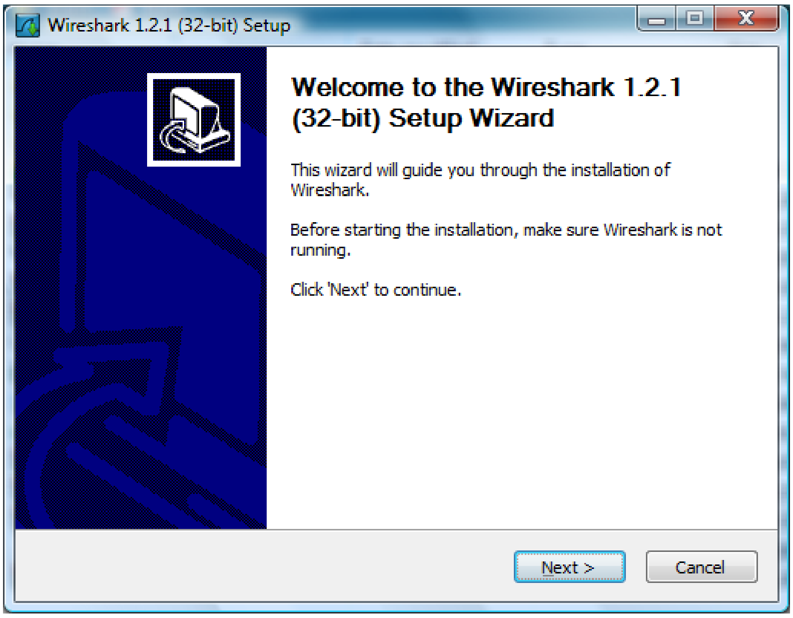
5.2 Using Wireshark to Capture and Analyse Traffic

In this exercise, the fundamentals of the **Wireshark Packet Sniffer and Protocol Analyser** tool will be introduced. Then Wireshark will be used to perform basic protocol analysis on TCP/IP network traffic.

The Wireshark User Guide can be found at:
http://www.wireshark.org/docs/wsug_html_chunked/

5.2.1 (Optional) Download and install Wireshark on your PC.

Wireshark is a network packet sniffer (and protocol analyzer) that runs on many platforms, including Windows XP and Vista. If Wireshark is not currently available on your PC, you can download the Latest Windows Version from [here] Wireshark 1.2.6 Windown Installer.
Other Versions of Wireshark from http://www.wireshark.org/download.html. The current version of Wireshark, at time of writing, is version 1.2.6. The initial Wireshark installation screen is shown in Figure 1.



Wireshark 1.2.1 (32-bit) Setup

Welcome to the Wireshark 1.2.1 (32-bit) Setup Wizard

This wizard will guide you through the installation of Wireshark.

Before starting the installation, make sure Wireshark is not running.

Click 'Next' to continue.

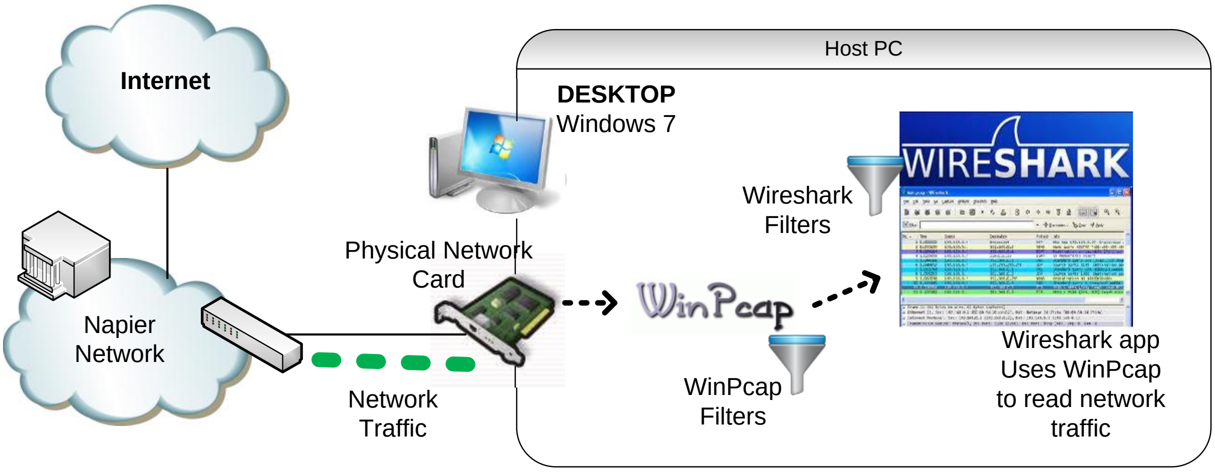
.

Figure 1 Wireshark Installation

Click the **I Agree** button to the License agreement, then select options (or accept defaults) clicking the **Next** button on each screen when prompted.

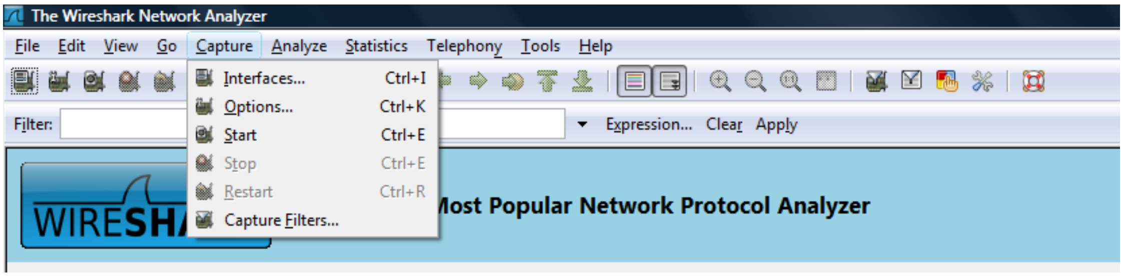
Note: On the **Install WinPcap?** window, select the install WinPcap options and select **Start WinPcap service** option, if you want to have other users besides those with administrative privileges to run Wireshark.

5.2.2 Using Wireshark to Capture Traffic



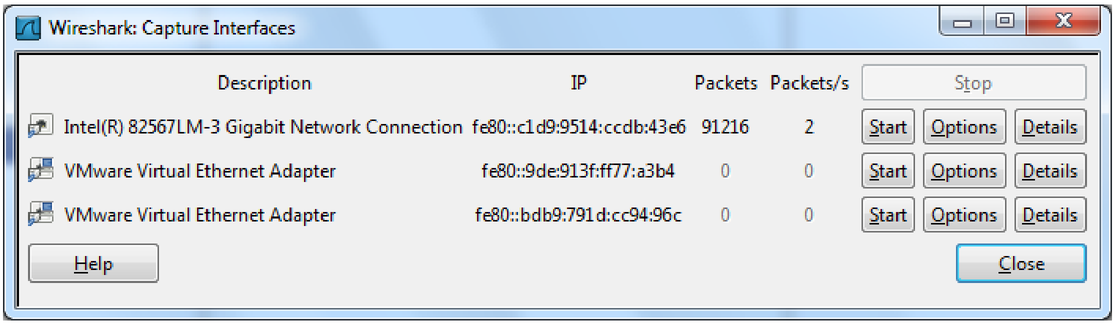
Select a Network Interface to Capture Packets through.

Start the Wireshark application. When Wireshark is first run, a default, or blank window is shown. To list the available network interfaces, select the **Capture->Interfaces** menu option.



The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Interfaces... Ctrl+I
Options... Ctrl+K
Start Ctrl+E
Stop Ctrl+E
Restart Ctrl+R
Capture Filters...
Filter:
Expression... Clear Apply
WIRESHA
Most Popular Network Protocol Analyzer

Wireshark should display a popup window such as the one shown in Figure 2. To capture network traffic click the **Start** button for the network interface you want to capture traffic on. Windows can have a long list of virtual interfaces, before the Ethernet Network Interface Card (NIC).



Wireshark: Capture Interfaces

Description	IP	Packets	Packets/s	Stop		
Intel(R) 82567LM-3 Gigabit Network Connection	fe80::c1d9:9514:ccdb:43e6	91216	2	Start	Options	Details
VMware Virtual Ethernet Adapter	fe80::9de:913f:ff77:a3b4	0	0	Start	Options	Details
VMware Virtual Ethernet Adapter	fe80::bdb9:791d:cc94:96c	0	0	Start	Options	Details

Figure 2 - Wireshark Interfaces Window

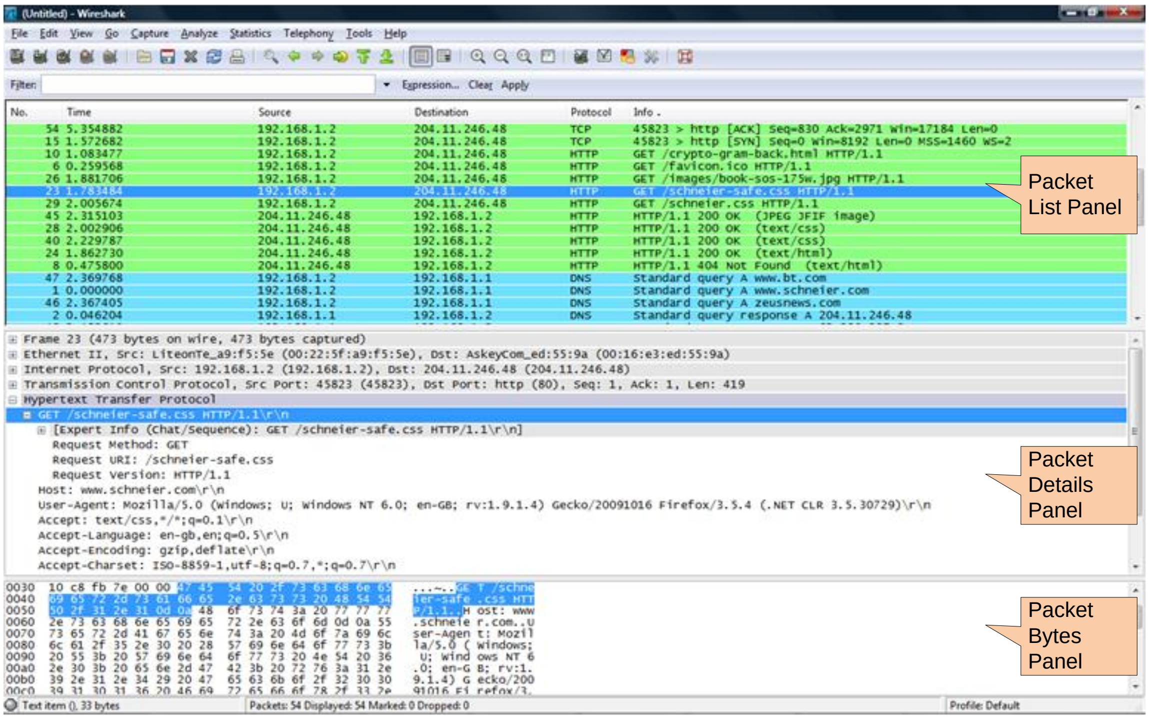
Questions

Q: Which Interface is connected to a local network (Ethernet)?

Q: How many packets have passed through the interface?

Note: The total incoming packets, for each interface, are displayed in the column to the left of the **Start** buttons.

Generate some network traffic with a Web Browser, such as Internet Explorer or Chrome. Your Wireshark window should show the packets, and now look something like.


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression... Clear Apply
Packet List Panel
Packet Details Panel
Packet Bytes Panel

```
No.   Time        Source          Destination     Protocol  Info
   54 5.354882    192.168.1.2     204.11.246.48   TCP       45823 > http [ACK] Seq=830 Ack=2971 Win=17184 Len=0
   15 1.572682    192.168.1.2     204.11.246.48   TCP       45823 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
   10 1.083477    192.168.1.2     204.11.246.48   HTTP      GET /crypto-gram-back.html HTTP/1.1
    6 0.259568    192.168.1.2     204.11.246.48   HTTP      GET /favicon.ico HTTP/1.1
   26 1.881706    192.168.1.2     204.11.246.48   HTTP      GET /images/book-sos-175w.jpg HTTP/1.1
   23 1.783484    192.168.1.2     204.11.246.48   HTTP      GET /schneier-safe.css HTTP/1.1
   29 2.005674    192.168.1.2     204.11.246.48   HTTP      GET /schneier.css HTTP/1.1
   45 2.315103    204.11.246.48   192.168.1.2     HTTP      HTTP/1.1 200 OK  (JPEG JFIF image)
   28 2.002906    204.11.246.48   192.168.1.2     HTTP      HTTP/1.1 200 OK  (text/css)
   40 2.229787    204.11.246.48   192.168.1.2     HTTP      HTTP/1.1 200 OK  (text/css)
   24 1.862730    204.11.246.48   192.168.1.2     HTTP      HTTP/1.1 200 OK  (text/html)
    8 0.475800    204.11.246.48   192.168.1.2     HTTP      HTTP/1.1 404 Not Found  (text/html)
   47 2.369768    192.168.1.2     192.168.1.1     DNS       Standard query A www.bt.com
    1 0.000000    192.168.1.2     192.168.1.1     DNS       Standard query A www.schneier.com
   46 2.367405    192.168.1.2     192.168.1.1     DNS       Standard query A zeusnews.com
    2 0.046204    192.168.1.1     192.168.1.2     DNS       Standard query response A 204.11.246.48
```

```
⊞ Frame 23 (473 bytes on wire, 473 bytes captured)
⊞ Ethernet II, Src: LiteonTe_a9:f5:5e (00:22:5f:a9:f5:5e), Dst: AskeyCom_ed:55:9a (00:16:e3:ed:55:9a)
⊞ Internet Protocol, Src: 192.168.1.2 (192.168.1.2), Dst: 204.11.246.48 (204.11.246.48)
⊞ Transmission Control Protocol, Src Port: 45823 (45823), Dst Port: http (80), Seq: 1, Ack: 1, Len: 419
⊟ Hypertext Transfer Protocol
  ⊟ GET /schneier-safe.css HTTP/1.1\r\n
    ⊞ [Expert Info (Chat/Sequence): GET /schneier-safe.css HTTP/1.1\r\n]
      Request Method: GET
      Request URI: /schneier-safe.css
      Request Version: HTTP/1.1
    Host: www.schneier.com\r\n
    User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; en-GB; rv:1.9.1.4) Gecko/20091016 Firefox/3.5.4 (.NET CLR 3.5.30729)\r\n
    Accept: text/css,*/*;q=0.1\r\n
    Accept-Language: en-gb,en;q=0.5\r\n
    Accept-Encoding: gzip,deflate\r\n
    Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
```

```
0030  10 c8 fb 7e 00 00 47 45  54 20 2f 73 63 68 6e 65   ...~..GE T /schne
0040  69 65 72 2d 73 61 66 65  2e 63 73 73 20 48 54 54   ier-safe .css HTT
0050  50 2f 31 2e 31 0d 0a 48  6f 73 74 3a 20 77 77 77   P/1.1..H ost: www
0060  2e 73 63 68 6e 65 69 65  72 2e 63 6f 6d 0d 0a 55   .schneie r.com..U
0070  73 65 72 2d 41 67 65 6e  74 3a 20 4d 6f 7a 69 6c   ser-Agen t: Mozil
0080  6c 61 2f 35 2e 30 20 28  57 69 6e 64 6f 77 73 3b   la/5.0 ( Windows;
0090  20 55 3b 20 57 69 6e 64  6f 77 73 20 4e 54 20 36    U; Wind ows NT 6
00a0  2e 30 3b 20 65 6e 2d 47  42 3b 20 72 76 3a 31 2e   .0; en-G B; rv:1.
00b0  39 2e 31 2e 34 29 20 47  65 63 6b 6f 2f 32 30 30   9.1.4) G ecko/200
00c0  39 31 30 31 36 20 46 69  72 65 66 6f 78 2f 33 2e   91016 Fi refox/3.
```

```
Text item (), 33 bytes    Packets: 54 Displayed: 54 Marked: 0 Dropped: 0    Profile: Default
```

Figure 3 - Wireshark capuring traffic

To stop the capture, select the **Capture->Stop** menu option, Ctrl+E, or the Stop toolbar button. What you have created is a Packet Capture or ***'pcap'***, which you can now view and analyse using the Wireshark interface, or save to disk to analyse later.

The capture is split into 3 parts:

1. **Packet List Panel –** this is a list of packets in the current capture. It colours the packets based on the protocol type. When a packet is selected, the details are shown in the two panels below.
2. **Packet Details Panel –** this shows the details of the selected packet. It shows the different protocols making up the layers of data for this packet. Layers include Frame, Ethernet, IP, TCP/UDP/ICMP, and application protocols such as HTTP.
3. **Packet Bytes Panel –** shows the packet bytes in Hex and ASCII encodings.

Questions

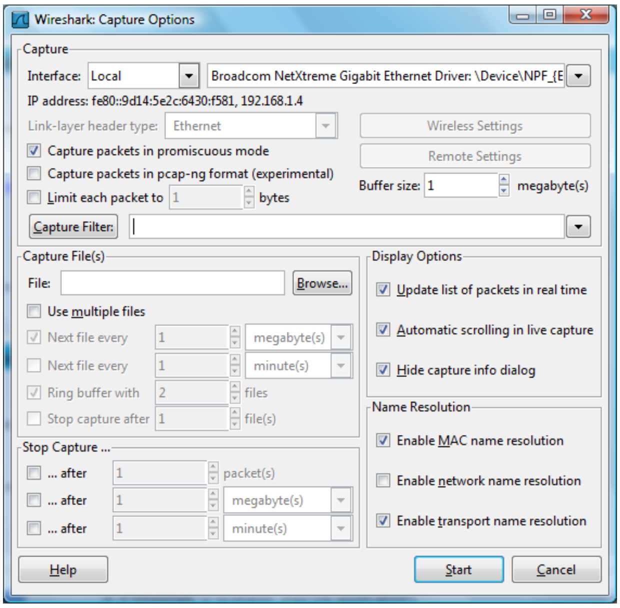
Search back through your capture, and find an **HTTP** packet containing a **GET** command. Click on the packet in the **Packet List Panel**. Then expand the HTTP layer in the **Packet Details Panel**, from the packet**.**

Q: From the **Packet Details Panel**, within the GET command, what is the value of the **Host**?

Q: Can you see the **Hex** and **ASCII** representations of the packet in the **Packet Bytes Panel**?

Q: From the **Packet Bytes Panel**, what are the first 4 bytes of the Hex value of the **Host** parameter?

To select more detailed options when starting a capture, select the **Capture->Options** menu option, or **Ctrl+K**, or the Capture Options button on the toolbar (the wrench). This should show a window such as shown in Figure 4.



Wireshark: Capture Options

Capture

Interface: Local | Broadcom NetXtreme Gigabit Ethernet Driver: \Device\NPF_{E

IP address: fe80::9d14:5e2c:6430:f581, 192.168.1.4

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)

Capture Filter:

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)

Stop Capture ...

☐ ... after 1 packet(s)

☐ ... after 1 megabyte(s)

☐ ... after 1 minute(s)

Display Options

☑ Update list of packets in real time

☑ Automatic scrolling in live capture

☑ Hide capture info dialog

Name Resolution

☑ Enable MAC name resolution

☐ Enable network name resolution

☑ Enable transport name resolution

Figure 4 - Wireshark Capture Options

Some of the more interesting options are:

- ***Capture Options > Interface*** **-** Again the important thing is to select the correct Network Interface to capture traffic through.
- ***Capture Options > Capture File*** **–** useful to save a file of the packet capture in real time, in case of a system crash.
- ***Display Options > Update list of packets in real time*** **–** A display option, which should be checked if you want to view the capture as it happens (typically switched off to capture straight to a file, for later analysis).
- ***Name Resolution > MAC name resolution*** **–** resolves the first 3 bytes of the MAC Address, the Organisation Unique Identifier (OUI), which represents the Manufacturer of the Card.
- ***Name Resolution > Network name resolution*** **–** does a DNS lookup for the IP Addresses captured, to display the network name. Set to off by default, so covert scans do not generate this DNS traffic, and tip off who's packets you are sniffing.

Make sure the **MAC name resolution** is selected. Start the capture, and generate some Web traffic again, then stop the capture.

Questions

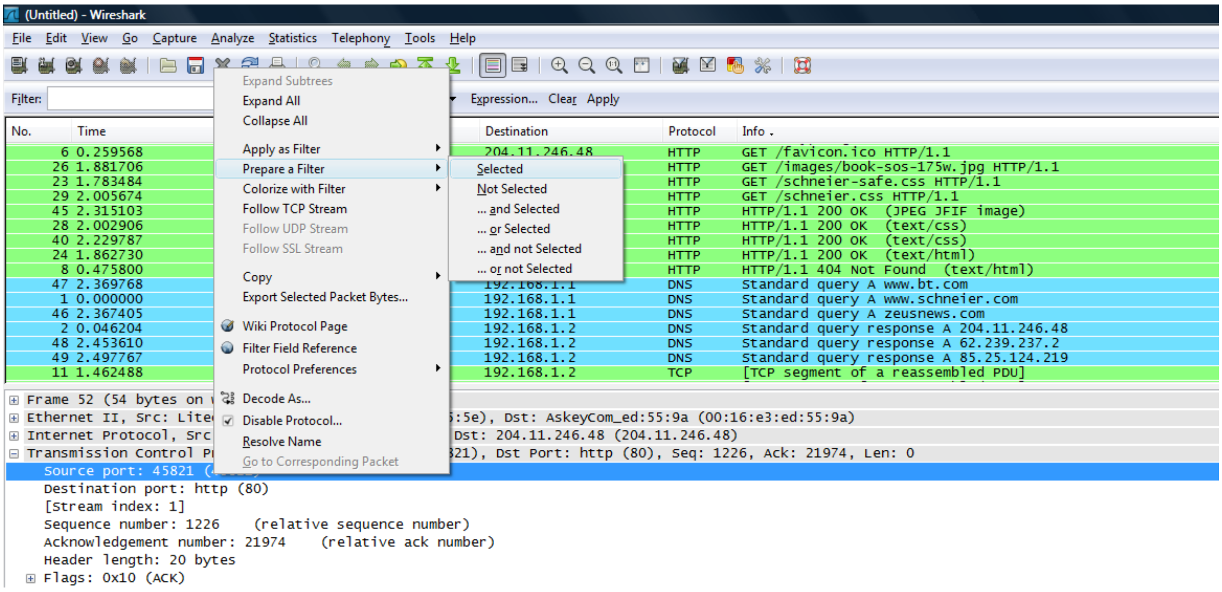
Search through your capture, and find an HTTP packet coming back from the server (TCP Source Port == 80). Expand the Ethernet layer in the **Packet Details Panel.**

Q. What are the manufacturers of your PC's Network Interface Card (NIC), and the servers NIC?

Q. What are the Hex values (shown the raw bytes panel) of the two NICS Manufacturers OUIs?

5.2.3 Wireshark Display Filters.

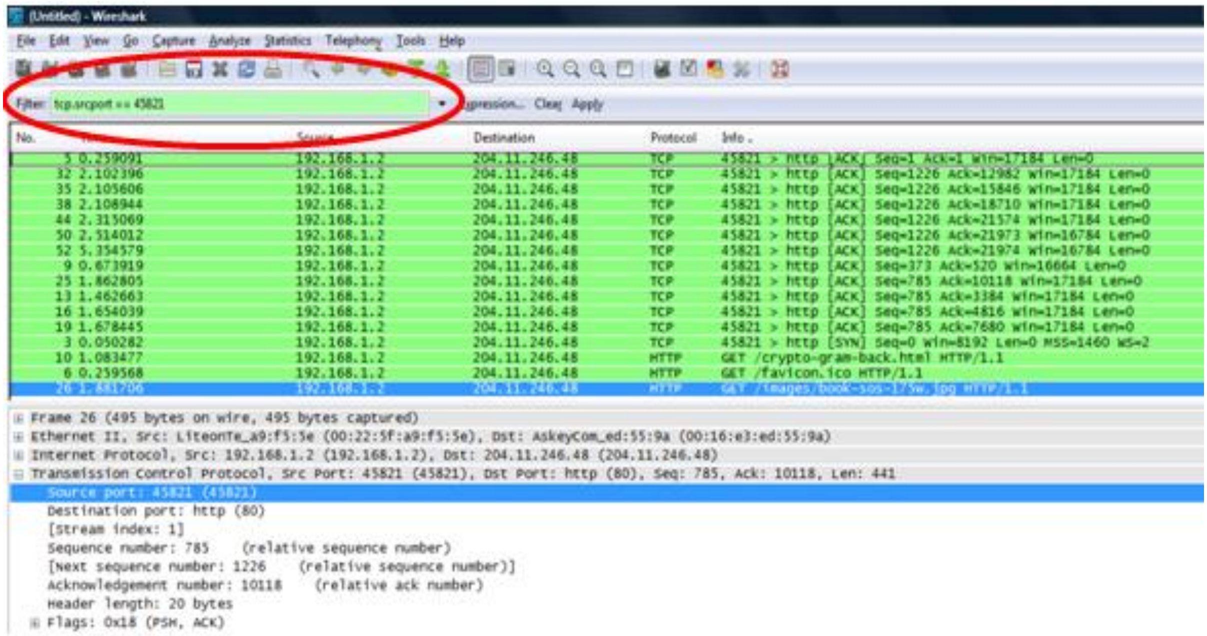
Right click on the **Source Port** field in the **Packet Details Panel**. Select **Prepare a Filter->Selected**.



```
No.  Time          Destination      Protocol  Info
  6 0.259568       204.11.246.48    HTTP      GET /favicon.ico HTTP/1.1
 26 1.881706                        HTTP      GET /images/book-sos-175w.jpg HTTP/1.1
 23 1.783484                        HTTP      GET /schneier-safe.css HTTP/1.1
 29 2.005674                        HTTP      GET /schneier.css HTTP/1.1
 45 2.315103                        HTTP      HTTP/1.1 200 OK  (JPEG JFIF image)
 28 2.002906                        HTTP      HTTP/1.1 200 OK  (text/css)
 40 2.229787                        HTTP      HTTP/1.1 200 OK  (text/css)
 24 1.862730                        HTTP      HTTP/1.1 200 OK  (text/html)
  8 0.475800                        HTTP      HTTP/1.1 404 Not Found  (text/html)
 47 2.369768       192.168.1.1      DNS       Standard query A www.bt.com
  1 0.000000       192.168.1.1      DNS       Standard query A www.schneier.com
 46 2.367405       192.168.1.1      DNS       Standard query A zeusnews.com
  2 0.046204       192.168.1.2      DNS       Standard query response A 204.11.246.48
 48 2.453610       192.168.1.2      DNS       Standard query response A 62.239.237.2
 49 2.497767       192.168.1.2      DNS       Standard query response A 85.25.124.219
 11 1.462488       192.168.1.2      TCP       [TCP segment of a reassembled PDU]

⊞ Frame 52 (54 bytes on 
⊞ Ethernet II, Src: Lite                  5:5e), Dst: AskeyCom_ed:55:9a (00:16:e3:ed:55:9a)
⊞ Internet Protocol, Src                  Dst: 204.11.246.48 (204.11.246.48)
⊟ Transmission Control Pr                 321), Dst Port: http (80), Seq: 1226, Ack: 21974, Len: 0
    Source port: 45821 (
    Destination port: http (80)
    [Stream index: 1]
    Sequence number: 1226    (relative sequence number)
    Acknowledgement number: 21974    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x10 (ACK)
```

Figure 5 - Filtering on a protocol field

Wireshark automatically generates a **Display Filter**, and applies it to the capture. The filter is shown in the **Filter Bar**, below the button toolbar. Only packets captured with a Source Port of the value selected should be displayed. The window should be similar to that shown in Figure 6. This same process can be performed on most fields within Wireshark, and can be used to include or exclude traffic.



(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Expression... Clear Apply

Filter: tcp.srcport == 45821

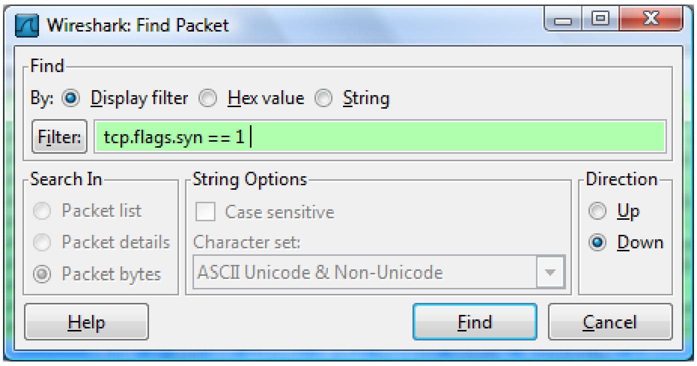
```
No.                    Source          Destination     Protocol  Info
     5 0.259091        192.168.1.2     204.11.246.48   TCP       45821 > http [ACK] Seq=1 Ack=1 Win=17184 Len=0
    32 2.102396        192.168.1.2     204.11.246.48   TCP       45821 > http [ACK] Seq=1226 Ack=12982 Win=17184 Len=0
    35 2.105606        192.168.1.2     204.11.246.48   TCP       45821 > http [ACK] Seq=1226 Ack=15846 Win=17184 Len=0
    38 2.108944        192.168.1.2     204.11.246.48   TCP       45821 > http [ACK] Seq=1226 Ack=18710 Win=17184 Len=0
    44 2.315069        192.168.1.2     204.11.246.48   TCP       45821 > http [ACK] Seq=1226 Ack=21574 Win=17184 Len=0
    50 2.514012        192.168.1.2     204.11.246.48   TCP       45821 > http [ACK] Seq=1226 Ack=21973 Win=16784 Len=0
    52 5.354579        192.168.1.2     204.11.246.48   TCP       45821 > http [ACK] Seq=1226 Ack=21974 Win=16784 Len=0
     9 0.673919        192.168.1.2     204.11.246.48   TCP       45821 > http [ACK] Seq=373 Ack=520 Win=16664 Len=0
    25 1.862805        192.168.1.2     204.11.246.48   TCP       45821 > http [ACK] Seq=785 Ack=10118 Win=17184 Len=0
    13 1.462663        192.168.1.2     204.11.246.48   TCP       45821 > http [ACK] Seq=785 Ack=3384 Win=17184 Len=0
    16 1.654039        192.168.1.2     204.11.246.48   TCP       45821 > http [ACK] Seq=785 Ack=4816 Win=17184 Len=0
    19 1.678445        192.168.1.2     204.11.246.48   TCP       45821 > http [ACK] Seq=785 Ack=7680 Win=17184 Len=0
     3 0.050282        192.168.1.2     204.11.246.48   TCP       45821 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
    10 1.083477        192.168.1.2     204.11.246.48   HTTP      GET /crypto-gram-back.html HTTP/1.1
     6 0.259568        192.168.1.2     204.11.246.48   HTTP      GET /favicon.ico HTTP/1.1
    26 1.881706        192.168.1.2     204.11.246.48   HTTP      GET /images/book-sos-175w.jpg HTTP/1.1

⊞ Frame 26 (495 bytes on wire, 495 bytes captured)
⊞ Ethernet II, Src: LiteonTe_a9:f5:5e (00:22:5f:a9:f5:5e), Dst: AskeyCom_ed:55:9a (00:16:e3:ed:55:9a)
⊞ Internet Protocol, Src: 192.168.1.2 (192.168.1.2), Dst: 204.11.246.48 (204.11.246.48)
⊟ Transmission Control Protocol, Src Port: 45821 (45821), Dst Port: http (80), Seq: 785, Ack: 10118, Len: 441
    Source port: 45821 (45821)
    Destination port: http (80)
    [Stream index: 1]
    Sequence number: 785    (relative sequence number)
    [Next sequence number: 1226    (relative sequence number)]
    Acknowledgement number: 10118    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x18 (PSH, ACK)
```

Figure 6 - Wireshark Display Filter

5.2.4 Analysing a TCP Session using Wireshark.

Start a capture, and generate some Web traffic by going to www.schneier.com, then stop the capture. Scroll back to the top of the capture trace. Find the first SYN packet, sent from your PC to the Web Server. This signifies the start of a TCP 3-way handshake.

If your having trouble finding the first SYN packet, select the **Edit->Find Packet** menu option. Select the **Display Filter** radio button and enter a filter of **tcp.flags.** (at this point you should get a list of the flags to choose from). Choose the correct flag, **tcp.flags.syn** and add **== 1**. Hit the **Find** button, and the first SYN packet in the trace should be highlighted.



By: ● Display filter ○ Hex value ○ String

Filter: tcp.flags.syn == 1

○ Packet list
○ Packet details
● Packet bytes

□ Case sensitive
Character set: ASCII Unicode & Non-Unicode

○ Up
● Down

Note: Find Packet can also be used to search for a Hex signature, such as a malware signature, or to search for a string – such as a protocol command - in the Packet Capture (**pcap**).

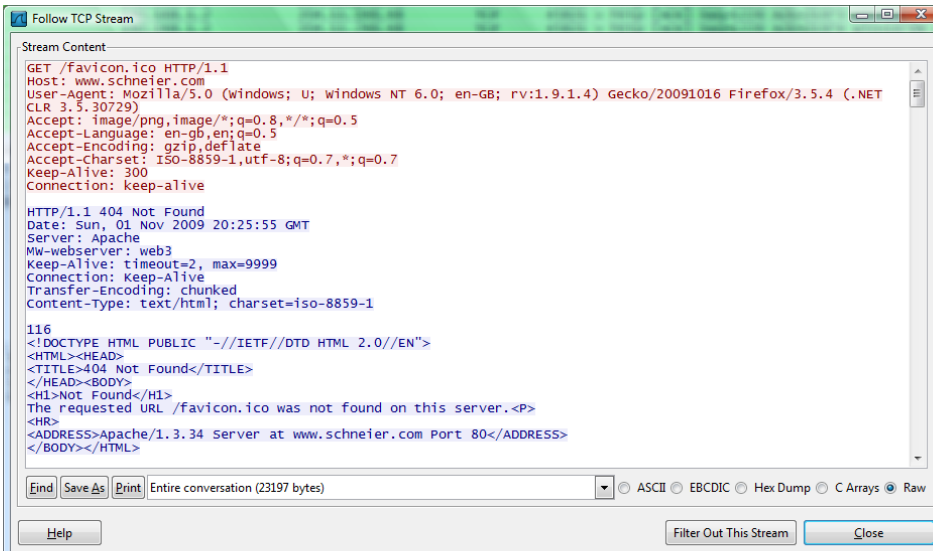
Questions

Q: Can you identify the rest of the TCP 3-way handshake easily? (if not read on)

A quick way to create a **Wireshark Display Filter** to isolate a TCP stream is to right click on a packet in the **Packet List Panel** and select **Follow TCP Stream.** This creates an automatic Display Filter which displays packets from that TCP session only.

It also pops up a session display window, containing by default, an ASCII representation of the TCP session with the client packets in red and the server packets in blue.

The window should look something like Figure 7. This is very useful for viewing human readable protocol payloads, such as HTTP, SMTP, and FTP.

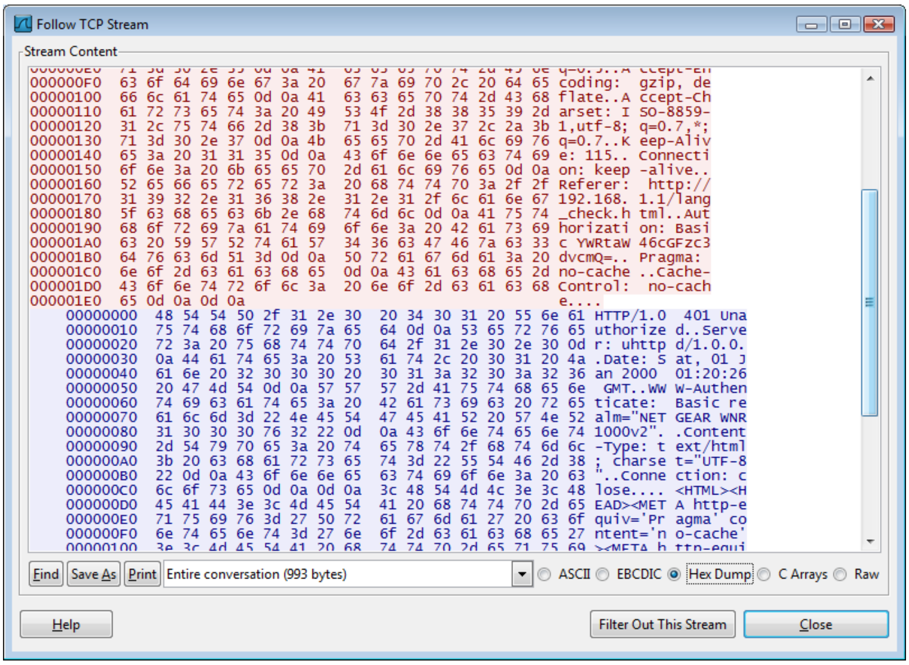


```
GET /favicon.ico HTTP/1.1
Host: www.schneier.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; en-GB; rv:1.9.1.4) Gecko/20091016 Firefox/3.5.4 (.NET
CLR 3.5.30729)
Accept: image/png,image/*;q=0.8,*/*;q=0.5
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive

HTTP/1.1 404 Not Found
Date: Sun, 01 Nov 2009 20:25:55 GMT
Server: Apache
MW-webserver: web3
Keep-Alive: timeout=2, max=9999
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1

116
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>404 Not Found</TITLE>
</HEAD><BODY>
<H1>Not Found</H1>
The requested URL /favicon.ico was not found on this server.<P>
<HR>
<ADDRESS>Apache/1.3.34 Server at www.schneier.com Port 80</ADDRESS>
</BODY></HTML>
```

Find | Save As | Print | Entire conversation (23197 bytes) ○ ASCII ○ EBCDIC ○ Hex Dump ○ C Arrays ● Raw

Figure 7 - Follow TCP Stream Window - ASCII

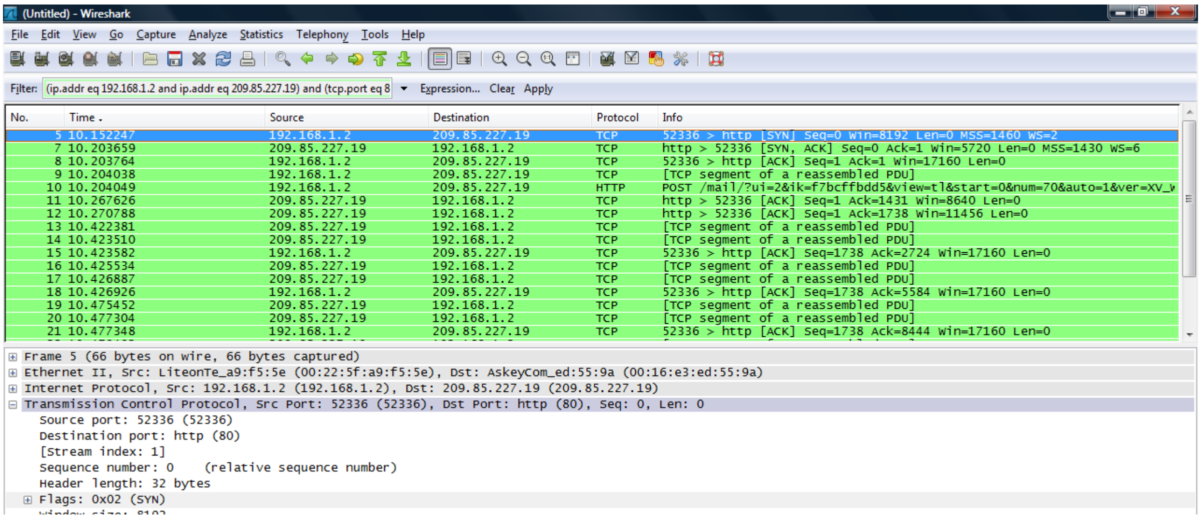
Change to Hex Dump Mode and view the payloads in raw Hex, as shown below.



```
000000E0  71 3d 30 2e 35 0d 0a 41  63 63 65 70 74 2d 45 6e q=0.5..A ccept-En
000000F0  63 6f 64 69 6e 67 3a 20  67 7a 69 70 2c 20 64 65 coding:  gzip, de
00000100  66 6c 61 74 65 0d 0a 41  63 63 65 70 74 2d 43 68 flate..A ccept-Ch
00000110  61 72 73 65 74 3a 20 49  53 4f 2d 38 38 35 39 2d arset: I SO-8859-
00000120  31 2c 75 74 66 2d 38 3b  71 3d 30 2e 37 2c 2a 3b 1,utf-8; q=0.7,*;
00000130  71 3d 30 2e 37 0d 0a 4b  65 65 70 2d 41 6c 69 76 q=0.7..K eep-Aliv
00000140  65 3a 20 31 31 35 0d 0a  43 6f 6e 6e 65 63 74 69 e: 115.. Connecti
00000150  6f 6e 3a 20 6b 65 65 70  2d 61 6c 69 76 65 0d 0a on: keep -alive..
00000160  52 65 66 65 72 65 72 3a  20 68 74 74 70 3a 2f 2f Referer:  http://
00000170  31 39 32 2e 31 36 38 2e  31 2e 31 2f 6c 61 6e 67 192.168. 1.1/lang
00000180  5f 63 68 65 63 6b 2e 68  74 6d 6c 0d 0a 41 75 74 _check.h tml..Aut
00000190  68 6f 72 69 7a 61 74 69  6f 6e 3a 20 42 61 73 69 horizati on: Basi
000001A0  63 20 59 57 52 74 61 57  34 36 63 47 46 7a 63 33 c YWRtaW 46cGFzc3
000001B0  64 76 63 6d 51 3d 0d 0a  50 72 61 67 6d 61 3a 20 dvcmQ=.. Pragma: 
000001C0  6e 6f 2d 63 61 63 68 65  0d 0a 43 61 63 68 65 2d no-cache ..Cache-
000001D0  43 6f 6e 74 72 6f 6c 3a  20 6e 6f 2d 63 61 63 68 Control:  no-cach
000001E0  65 0d 0a 0d 0a                                   e....
    00000000  48 54 54 50 2f 31 2e 30  20 34 30 31 20 55 6e 61 HTTP/1.0  401 Una
    00000010  75 74 68 6f 72 69 7a 65  64 0d 0a 53 65 72 76 65 uthorize d..Serve
    00000020  72 3a 20 75 68 74 74 70  64 2f 31 2e 30 2e 30 0d r: uhttp d/1.0.0.
    00000030  0a 44 61 74 65 3a 20 53  61 74 2c 20 30 31 20 4a .Date: S at, 01 J
    00000040  61 6e 20 32 30 30 30 20  30 31 3a 32 30 3a 32 36 an 2000  01:20:26
    00000050  20 47 4d 54 0d 0a 57 57  57 2d 41 75 74 68 65 6e  GMT..WW W-Authen
    00000060  74 69 63 61 74 65 3a 20  42 61 73 69 63 20 72 65 ticate:  Basic re
    00000070  61 6c 6d 3d 22 4e 45 54  47 45 41 52 20 57 4e 52 alm="NET GEAR WNR
    00000080  31 30 30 30 76 32 22 0d  0a 43 6f 6e 74 65 6e 74 1000v2". .Content
    00000090  2d 54 79 70 65 3a 20 74  65 78 74 2f 68 74 6d 6c -Type: t ext/html
    000000A0  3b 20 63 68 61 72 73 65  74 3d 22 55 54 46 2d 38 ; charse t="UTF-8
    000000B0  22 0d 0a 43 6f 6e 6e 65  63 74 69 6f 6e 3a 20 63 "..Conne ction: c
    000000C0  6c 6f 73 65 0d 0a 0d 0a  3c 48 54 4d 4c 3e 3c 48 lose.... <HTML><H
    000000D0  45 41 44 3e 3c 4d 45 54  41 20 68 74 74 70 2d 65 EAD><MET A http-e
    000000E0  71 75 69 76 3d 27 50 72  61 67 6d 61 27 20 63 6f quiv='Pr agma' co
    000000F0  6e 74 65 6e 74 3d 27 6e  6f 2d 63 61 63 68 65 27 ntent='n o-cache'
    00000100  3e 3c 4d 45 54 41 20 68  74 74 70 2d 65 71 75 69 ><META h ttp-equi
```

Find | Save As | Print | Entire conversation (993 bytes) ○ ASCII ○ EBCDIC ● Hex Dump ○ C Arrays ○ Raw

Close the popup window. Wireshark now only shows the packets from the selected TCP Stream. You should be able to identify the 3-way handshake easily now.


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Expression... Clear Apply

Filter: (ip.addr eq 192.168.1.2 and ip.addr eq 209.85.227.19) and (tcp.port eq 8

```
No.   Time .          Source           Destination      Protocol  Info
    5 10.152247       192.168.1.2      209.85.227.19    TCP       52336 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
    7 10.203659       209.85.227.19    192.168.1.2      TCP       http > 52336 [SYN, ACK] Seq=0 Ack=1 Win=5720 Len=0 MSS=1430 WS=6
    8 10.203764       192.168.1.2      209.85.227.19    TCP       52336 > http [ACK] Seq=1 Ack=1 Win=17160 Len=0
    9 10.204038       192.168.1.2      209.85.227.19    TCP       [TCP segment of a reassembled PDU]
   10 10.204049       192.168.1.2      209.85.227.19    HTTP      POST /mail/?ui=2&ik=f7bcffbdd5&view=tl&start=0&num=70&auto=1&ver=XV_W
   11 10.267626       209.85.227.19    192.168.1.2      TCP       http > 52336 [ACK] Seq=1 Ack=1431 Win=8640 Len=0
   12 10.270788       209.85.227.19    192.168.1.2      TCP       http > 52336 [ACK] Seq=1 Ack=1738 Win=11456 Len=0
   13 10.422381       209.85.227.19    192.168.1.2      TCP       [TCP segment of a reassembled PDU]
   14 10.423510       209.85.227.19    192.168.1.2      TCP       [TCP segment of a reassembled PDU]
   15 10.423582       192.168.1.2      209.85.227.19    TCP       52336 > http [ACK] Seq=1738 Ack=2724 Win=17160 Len=0
   16 10.425534       209.85.227.19    192.168.1.2      TCP       [TCP segment of a reassembled PDU]
   17 10.426887       209.85.227.19    192.168.1.2      TCP       [TCP segment of a reassembled PDU]
   18 10.426926       192.168.1.2      209.85.227.19    TCP       52336 > http [ACK] Seq=1738 Ack=5584 Win=17160 Len=0
   19 10.475452       209.85.227.19    192.168.1.2      TCP       [TCP segment of a reassembled PDU]
   20 10.477304       209.85.227.19    192.168.1.2      TCP       [TCP segment of a reassembled PDU]
   21 10.477348       192.168.1.2      209.85.227.19    TCP       52336 > http [ACK] Seq=1738 Ack=8444 Win=17160 Len=0
```

```
⊞ Frame 5 (66 bytes on wire, 66 bytes captured)
⊞ Ethernet II, Src: LiteonTe_a9:f5:5e (00:22:5f:a9:f5:5e), Dst: AskeyCom_ed:55:9a (00:16:e3:ed:55:9a)
⊞ Internet Protocol, Src: 192.168.1.2 (192.168.1.2), Dst: 209.85.227.19 (209.85.227.19)
⊟ Transmission Control Protocol, Src Port: 52336 (52336), Dst Port: http (80), Seq: 0, Len: 0
    Source port: 52336 (52336)
    Destination port: http (80)
    [Stream index: 1]
    Sequence number: 0    (relative sequence number)
    Header length: 32 bytes
  ⊞ Flags: 0x02 (SYN)
```

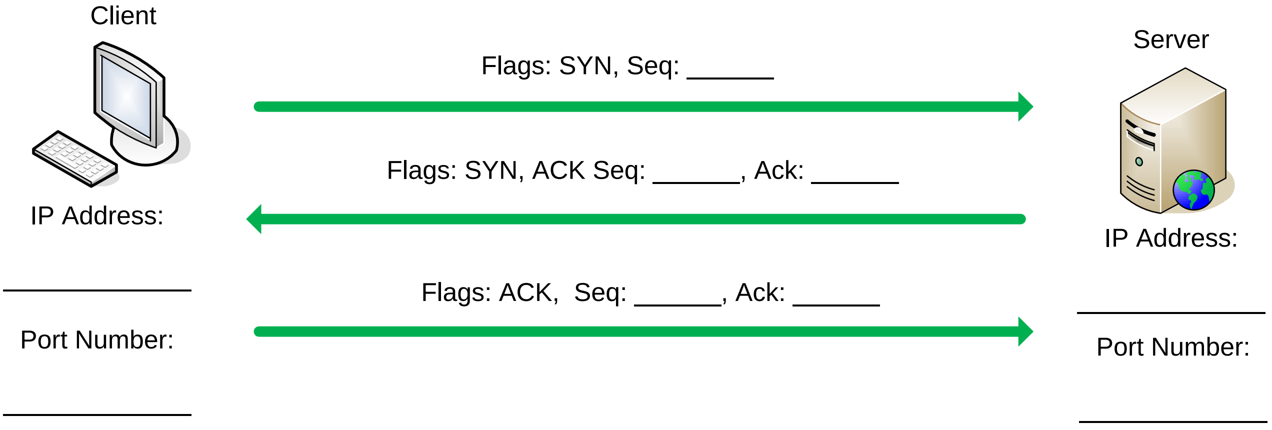
Note: Wireshark has automatically created a **display filter** to filter out this TCP conversation. In this case:

(ip.addr eq 192.168.1.2 and ip.addr eq 209.85.227.19) and (tcp.port eq 80 and tcp.port eq 52336)

Questions

Q. From your Wireshark Capture, fill in the diagram below with the IP Addresses and Port Numbers for the Client and the Server

Q. For each packet in the TCP 3-way handshake, fill in the Sequence and Acknowledgement numbers, on the diagram below.



5.2.5 Saving Packet Captures

Often captures should be saved to disc, for later analysis. To save a capture, select **File->Save As** and save the trace. By default this creates a Wireshark **pcapng** file, or if you select **pcap** a file many tools can read and write this. For example a tcpdump output file is in this format and can be read into Wireshark for analysis. This saves all the captured packets to the file.

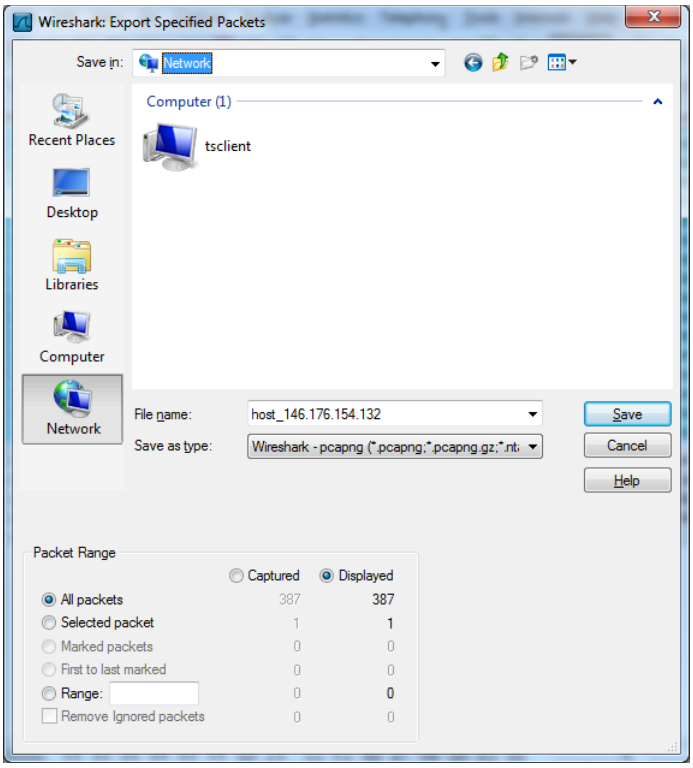
Questions

Q: Did you successfully save your capture to disc?

Q: Copy the **Display Filter** into the clipboard, and close and start Wireshark again, then reload the file. Was the whole capture saved or just the displayed packets?

Paste the display filter back into the Filter Bar, and Apply it.

To save *only the displayed packets*, select **File-> Export Specified Packets**, and make sure the **Displayed** radio button is selected rather than the **Captured** option. This creates a **pcap** file, with only the packets filtered by the current display filter.



Save in: Network

File name: host_146.176.154.132

Save as type: Wireshark - pcapng (*.pcapng;*.pcapng.gz;*.nti

Packet Range

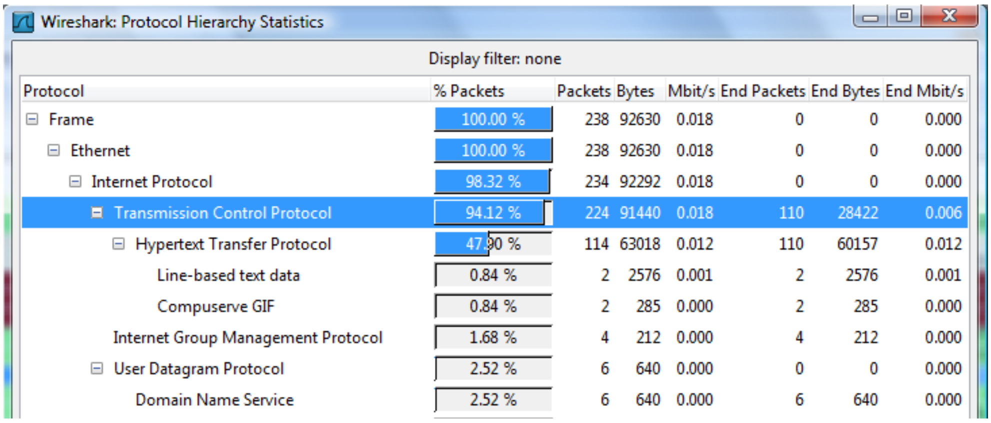
	○ Captured	● Displayed
● All packets	387	387
○ Selected packet	1	1
○ Marked packets	0	0
○ First to last marked	0	0
○ Range:	0	0
□ Remove Ignored packets	0	0

Questions

Q: Close and start Wireshark again, then reload the file. Was the whole capture saved or just the displayed packets?

5.2.6 Wireshark Statistics

Start the capture, and generate some Web traffic by going to www.schneier.com, then stop the capture, and select the **Statistics->Protocol Hierarchy** menu option. A window similar to that shown in Figure 8 should be shown displaying statictics about the **pcap**. Note that all the packets are Ethernet (Local Area Network) packets, but at the network layer most of the packets are TCP, but some are UDP.



Display filter: none

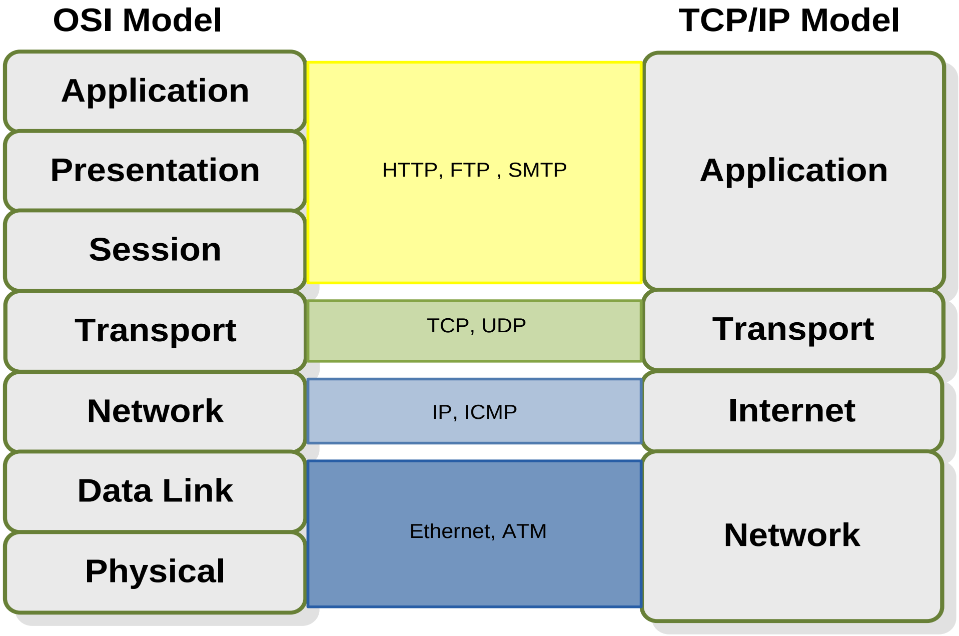
Protocol	% Packets	Packets	Bytes	Mbit/s	End Packets	End Bytes	End Mbit/s
⊟ Frame	100.00 %	238	92630	0.018	0	0	0.000
⊟ Ethernet	100.00 %	238	92630	0.018	0	0	0.000
⊟ Internet Protocol	98.32 %	234	92292	0.018	0	0	0.000
⊟ Transmission Control Protocol	94.12 %	224	91440	0.018	110	28422	0.006
⊟ Hypertext Transfer Protocol	47.90 %	114	63018	0.012	110	60157	0.012
Line-based text data	0.84 %	2	2576	0.001	2	2576	0.001
Compuserve GIF	0.84 %	2	285	0.000	2	285	0.000
Internet Group Management Protocol	1.68 %	4	212	0.000	4	212	0.000
⊟ User Datagram Protocol	2.52 %	6	640	0.000	0	0	0.000
Domain Name Service	2.52 %	6	640	0.000	6	640	0.000

Figure 8 - Wireshark Statistics Window

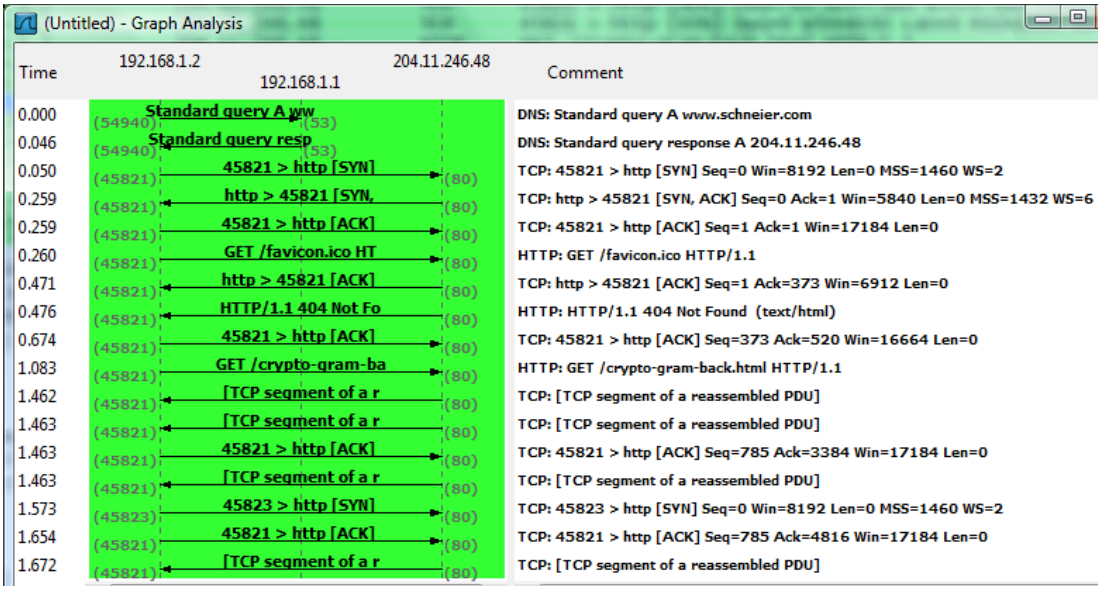
Questions

Q: What percentage of packets in your capture are TCP, and give an example of the higher level protocol which uses TCP?

Q: What percentage of packets in your capture are UDP, and give an example of the higher level protocol which uses UDP? (use the figure below)



Select the **Statistics->Flow Graph** menu option. Choose **General Flow** and **Network Source** options, and click the **OK** button. A window similar to that shown in should be displayed, showing the flow of traffic.

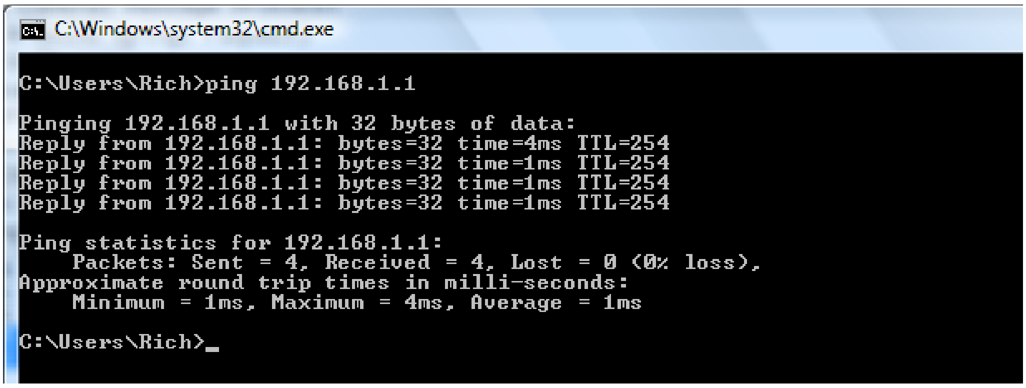


```
0.000  DNS: Standard query A www.schneier.com
0.046  DNS: Standard query response A 204.11.246.48
0.050  TCP: 45821 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
0.259  TCP: http > 45821 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1432 WS=6
0.259  TCP: 45821 > http [ACK] Seq=1 Ack=1 Win=17184 Len=0
0.260  HTTP: GET /favicon.ico HTTP/1.1
0.471  TCP: http > 45821 [ACK] Seq=1 Ack=373 Win=6912 Len=0
0.476  HTTP: HTTP/1.1 404 Not Found  (text/html)
0.674  TCP: 45821 > http [ACK] Seq=373 Ack=520 Win=16664 Len=0
1.083  HTTP: GET /crypto-gram-back.html HTTP/1.1
1.462  TCP: [TCP segment of a reassembled PDU]
1.463  TCP: [TCP segment of a reassembled PDU]
1.463  TCP: 45821 > http [ACK] Seq=785 Ack=3384 Win=17184 Len=0
1.463  TCP: [TCP segment of a reassembled PDU]
1.573  TCP: 45823 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
1.654  TCP: 45821 > http [ACK] Seq=785 Ack=4816 Win=17184 Len=0
1.672  TCP: [TCP segment of a reassembled PDU]
```

Figure 9 - Wireshark Flow Graph Window

5.2.7 Capture ARP & ICMP Protocol Traffic using Wireshark.

Start a Wireshark capture. Open a Windows console window, and generate some ICMP traffic by using the **`Ping`** command line tool to check the connectivity of a neighbouring machine (or your home router).

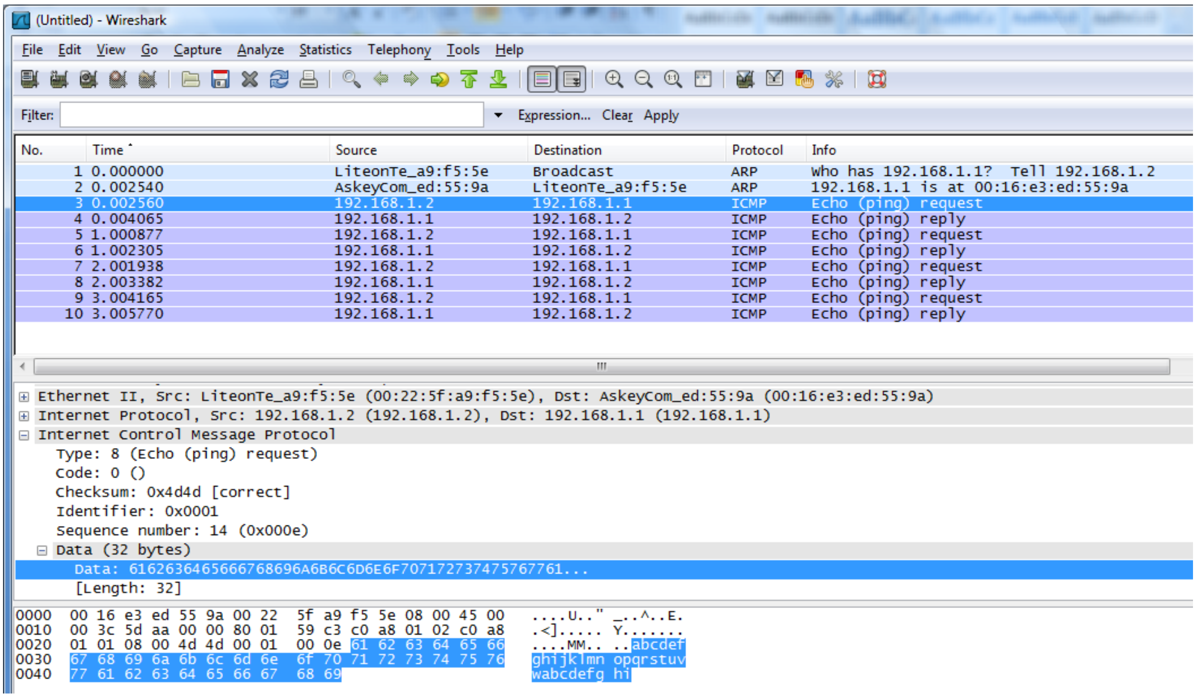


```
C:\Users\Rich>ping 192.168.1.1

Pinging 192.168.1.1 with 32 bytes of data:
Reply from 192.168.1.1: bytes=32 time=4ms TTL=254
Reply from 192.168.1.1: bytes=32 time=1ms TTL=254
Reply from 192.168.1.1: bytes=32 time=1ms TTL=254
Reply from 192.168.1.1: bytes=32 time=1ms TTL=254

Ping statistics for 192.168.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 4ms, Average = 1ms

C:\Users\Rich>_
```

Stop the capture and Wireshark should now look something like Figure 10.

The Address Resoloution Protocol (ARP) and ICMP packets are difficult to pick out, create a **display filter** to only show ARP or ICMP packets.

Some useful Wireshark **display filters** can be found at:
http://wiki.wireshark.org/DisplayFilters


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Info
1	0.000000	LiteonTe_a9:f5:5e	Broadcast	ARP	Who has 192.168.1.1? Tell 192.168.1.2
2	0.002540	AskeyCom_ed:55:9a	LiteonTe_a9:f5:5e	ARP	192.168.1.1 is at 00:16:e3:ed:55:9a
3	0.002560	192.168.1.2	192.168.1.1	ICMP	Echo (ping) request
4	0.004065	192.168.1.1	192.168.1.2	ICMP	Echo (ping) reply
5	1.000877	192.168.1.2	192.168.1.1	ICMP	Echo (ping) request
6	1.002305	192.168.1.1	192.168.1.2	ICMP	Echo (ping) reply
7	2.001938	192.168.1.2	192.168.1.1	ICMP	Echo (ping) request
8	2.003382	192.168.1.1	192.168.1.2	ICMP	Echo (ping) reply
9	3.004165	192.168.1.2	192.168.1.1	ICMP	Echo (ping) request
10	3.005770	192.168.1.1	192.168.1.2	ICMP	Echo (ping) reply

```
⊞ Ethernet II, Src: LiteonTe_a9:f5:5e (00:22:5f:a9:f5:5e), Dst: AskeyCom_ed:55:9a (00:16:e3:ed:55:9a)
⊞ Internet Protocol, Src: 192.168.1.2 (192.168.1.2), Dst: 192.168.1.1 (192.168.1.1)
⊟ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0 ()
    Checksum: 0x4d4d [correct]
    Identifier: 0x0001
    Sequence number: 14 (0x000e)
  ⊟ Data (32 bytes)
      Data: 6162636465666768696A6B6C6D6E6F7071727374757677761...
      [Length: 32]
```

```
0000  00 16 e3 ed 55 9a 00 22  5f a9 f5 5e 08 00 45 00   ....U.." _..^..E.
0010  00 3c 5d aa 00 00 80 01  59 c3 c0 a8 01 02 c0 a8   .<]..... Y.......
0020  01 01 08 00 4d 4d 00 01  00 0e 61 62 63 64 65 66   ....MM.. ..abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```

Figure 10 - Wireshark showing ICMP packets

Note the results in Wireshark. The initial ARP request broadcast from your PC determines the physical MAC address of the network IP Address 192.168.1.1, and the ARP reply from the neighbouring system. After the ARP request, the pings (ICMP echo request and replies) can be seen.

Questions

Q. After the first ping command, are the ARP and ICMP packets captured by WIreshark?

YES/NO

Q. After a second or third ping command, is the ARP and ICMP packets captured by WIreshark?

ARP: YES/NO

ICMP: YES/NO

Q. Why is this?

If pinging the same system more than once, delete the ARP cache on your system, using the `arp` command, as shoen below, so an new ARP request will be generated.

```
C:\> ping 192.168.1.1
... ping output ...
C:\> arp -d *
```

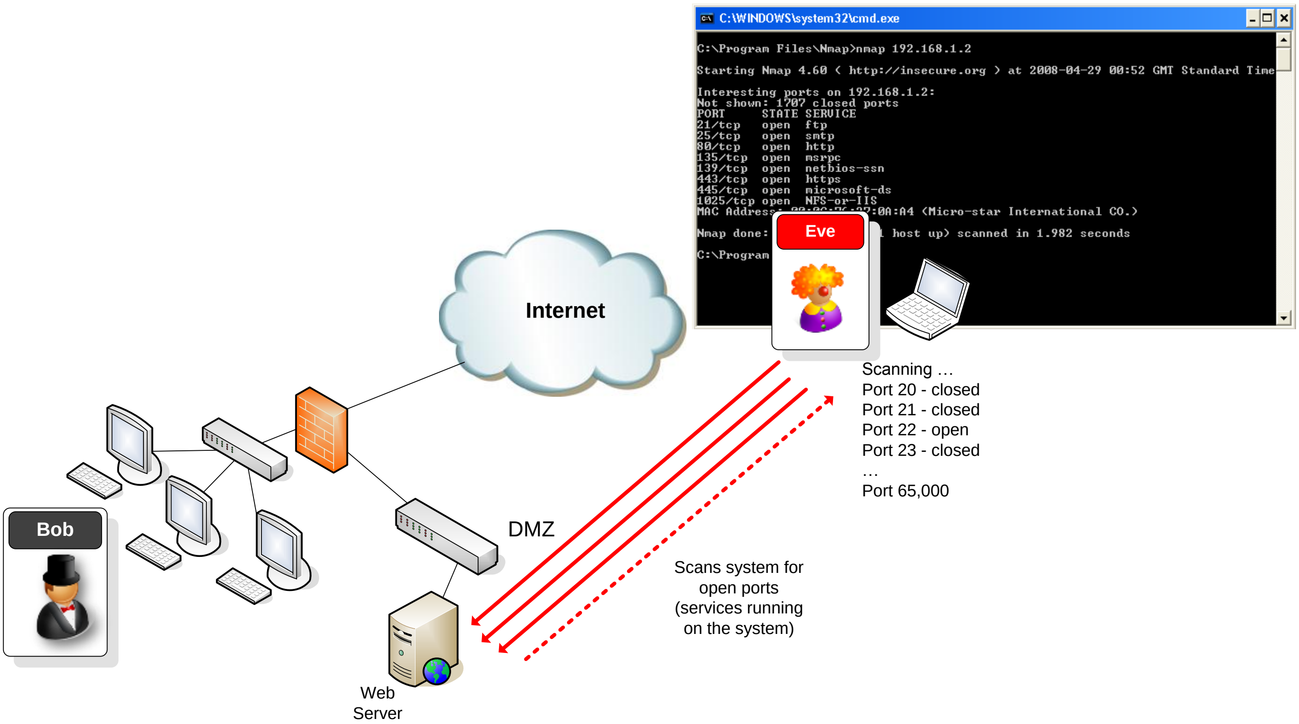
Note the results in Wireshark. The initial ARP request broadcast from your PC determines the physical MAC address of the network IP Address 192.168.1.1, and the ARP reply from the neighbouring system. After the ARP request, the pings (ICMP echo request and replies) can be seen.

5.2.8 Network Scanning

Network scanning is done at the **reconnaissance** stage of a structured attack. A network scanner can provide an attacker with information on remote machines which are alive, and that the attacker can communicate with, as well as the services those systems are running. Scanning includes **host sweeps/scans**, **OS scans**, **port scans** and **ping sweeps/scans**.

A **host scan** is typically done over an entire network, and reports machines which are alive on the network. A **port scan** is performed on a single, remote, host system, via its IP Address, and gives information on services running on the machine. Typically an attacker is also looking for which OS the system is running as well as any open TCP and UDP ports (services) which the attacker may be able to exploit. A network scanning tool, such as **nmap,** can be used to automatically probe the system for open ports, and give a report back to the attacker.

To mitigate open ports which attackers could use to compromise the system, make sure only services which are necessary are running. Some server OSs have services running by default, such as HTTP (port 80) and FTP (ports 20 & 21) which should be removed when systems are installed. (The command line network utility **netstat** can be used to check which services are running on the same host).



```
C:\Program Files\Nmap>nmap 192.168.1.2

Starting Nmap 4.60 ( http://insecure.org ) at 2008-04-29 00:52 GMT Standard Time

Interesting ports on 192.168.1.2:
Not shown: 1707 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
25/tcp   open  smtp
80/tcp   open  http
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
443/tcp  open  https
445/tcp  open  microsoft-ds
1025/tcp open  NFS-or-IIS
MAC Address: [illegible]:0A:A4 (Micro-star International CO.)

Nmap done: [illegible] host up) scanned in 1.982 seconds

C:\Program
```

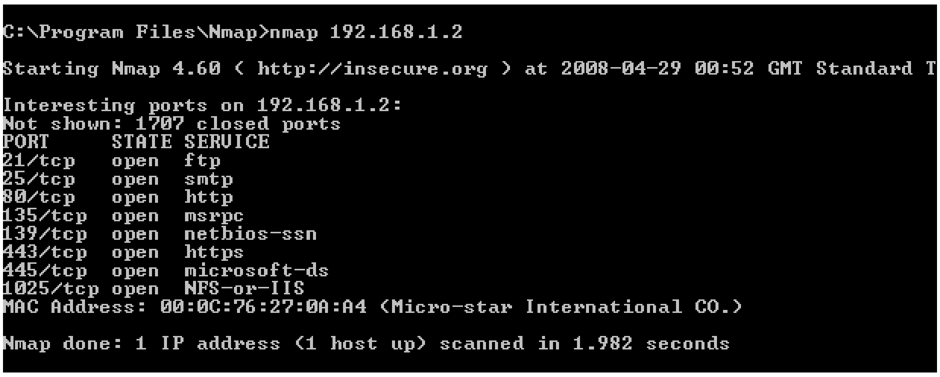
Figure 11 Port Scanning a System

5.2.9 Using Wireshark to Capture Network Scanning Traffic

If **nmap** is not on your system, download and install from:
http://nmap.org/dist/nmap-5.21-setup.exe

The **nmap** manual is available:
http://nmap.org/book/man.html

Start a Wireshark capture. Open a Windows command window, and perform a **Host Scan** (using ICMP packets) on a neighbours machine using **`nmap -sP [neighbours ip address]`** (do not scan the entire subnet). Stop the capture and filter the traffic for ARP and ICMP packets if necessary. Compare the capture with the saved ICMP capture from the previous section.



```
C:\Program Files\Nmap>nmap 192.168.1.2

Starting Nmap 4.60 ( http://insecure.org ) at 2008-04-29 00:52 GMT Standard T

Interesting ports on 192.168.1.2:
Not shown: 1707 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
25/tcp   open  smtp
80/tcp   open  http
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
443/tcp  open  https
445/tcp  open  microsoft-ds
1025/tcp open  NFS-or-IIS
MAC Address: 00:0C:76:27:0A:A4 (Micro-star International CO.)

Nmap done: 1 IP address (1 host up) scanned in 1.982 seconds
```

Figure 12 Nmap command line reconnaissance tool

Questions

Q: Are the packets the same as the **ping** packets from the capture in the previous section?

Q: What type of packets are sent by **nmap**?

Q: Are any other packets sent by **nmap**, during the ICMP probe?

Start a new Wireshark capture, and then perform a **host scan** (ICMP scan) on a system outwith the subnet, such as

```
nmap -sP scanme.nmap.org
```

(do not perform any other type of scan outside the lab subnet). Stop the capture and filter the traffic for ARP and ICMP packets if necessary. Compare the capture with the saved ICMP capture from section 6.

Questions

Q: Are the packets the same as the **ping** packets from the capture in section 6?

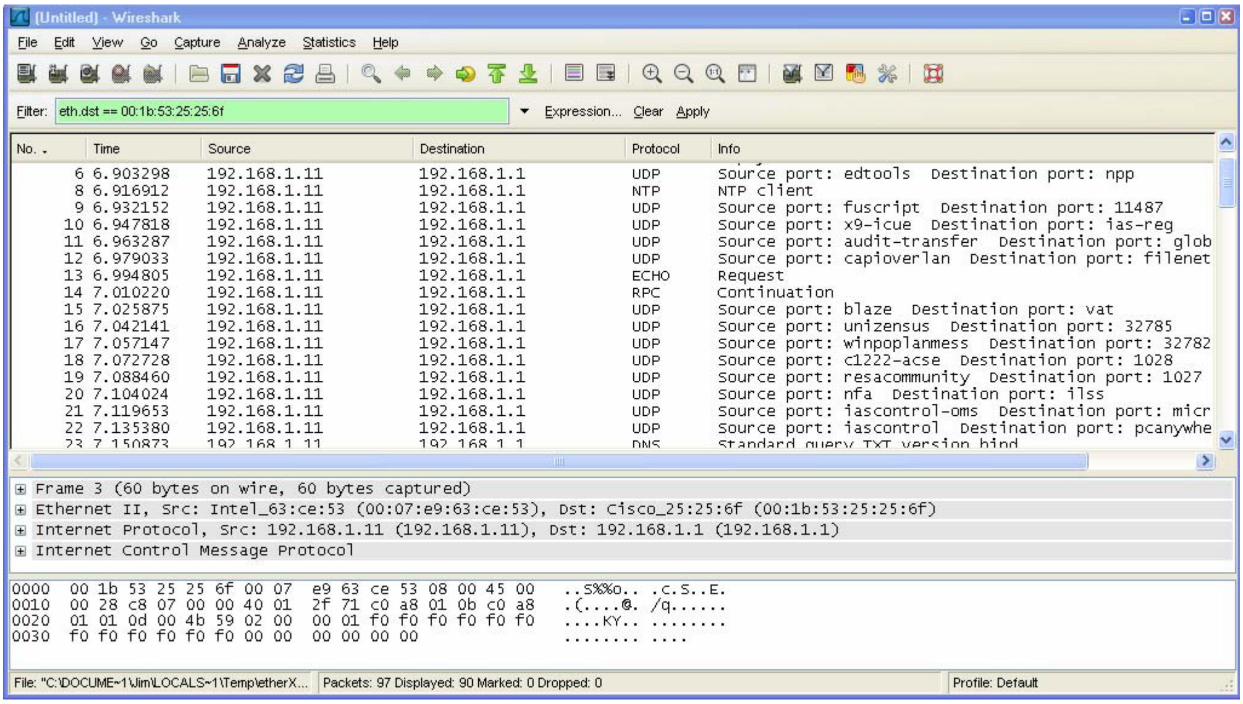
Q: What different types of packets are sent by **nmap**?

Q: Are any other packets sent by **nmap**, during the host scan? Which protocol and to which port?

Start a new Wireshark capture, and then perform a complete **Port Scan** (in this case a TCP SYN scan) and an **Operating System Fingerprint** on a neighbours machine using

```
nmap -O [neighbours ip address]
```

(do not scan more than a single machine). The `-O` option should provide the OS running on the scanned machine. Stop the capture and filter for source address == your machines address if necessary. Notice the number and types of ports tried by the nmap port scan. The capture should look something like Figure 13.


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Expression... Clear Apply

Filter: eth.dst == 00:1b:53:25:25:6f

```
No. .  Time        Source          Destination    Protocol  Info
     6 6.903298    192.168.1.11    192.168.1.1    UDP       Source port: edtools  Destination port: npp
     8 6.916912    192.168.1.11    192.168.1.1    NTP       NTP client
     9 6.932152    192.168.1.11    192.168.1.1    UDP       Source port: fuscript  Destination port: 11487
    10 6.947818    192.168.1.11    192.168.1.1    UDP       Source port: x9-icue  Destination port: ias-reg
    11 6.963287    192.168.1.11    192.168.1.1    UDP       Source port: audit-transfer  Destination port: glob
    12 6.979033    192.168.1.11    192.168.1.1    UDP       Source port: capioverlan  Destination port: filenet
    13 6.994805    192.168.1.11    192.168.1.1    ECHO      Request
    14 7.010220    192.168.1.11    192.168.1.1    RPC       Continuation
    15 7.025875    192.168.1.11    192.168.1.1    UDP       Source port: blaze  Destination port: vat
    16 7.042141    192.168.1.11    192.168.1.1    UDP       Source port: unizensus  Destination port: 32785
    17 7.057147    192.168.1.11    192.168.1.1    UDP       Source port: winpoplanmess  Destination port: 32782
    18 7.072728    192.168.1.11    192.168.1.1    UDP       Source port: c1222-acse  Destination port: 1028
    19 7.088460    192.168.1.11    192.168.1.1    UDP       Source port: resacommunity  Destination port: 1027
    20 7.104024    192.168.1.11    192.168.1.1    UDP       Source port: nfa  Destination port: ilss
    21 7.119653    192.168.1.11    192.168.1.1    UDP       Source port: iascontrol-oms  Destination port: micr
    22 7.135380    192.168.1.11    192.168.1.1    UDP       Source port: iascontrol  Destination port: pcanywhe
    23 7.150873    192.168.1.11    192.168.1.1    DNS       Standard query TXT version bind

⊞ Frame 3 (60 bytes on wire, 60 bytes captured)
⊞ Ethernet II, Src: Intel_63:ce:53 (00:07:e9:63:ce:53), Dst: Cisco_25:25:6f (00:1b:53:25:25:6f)
⊞ Internet Protocol, Src: 192.168.1.11 (192.168.1.11), Dst: 192.168.1.1 (192.168.1.1)
⊞ Internet Control Message Protocol

0000  00 1b 53 25 25 6f 00 07  e9 63 ce 53 08 00 45 00   ..S%%o.. .c.S..E.
0010  00 28 c8 07 00 00 40 01  2f 71 c0 a8 01 0b c0 a8   .(....@. /q......
0020  01 01 0d 00 4b 59 02 00  00 01 f0 f0 f0 f0 f0 f0   ....KY.. ........
0030  f0 f0 f0 f0 f0 f0 00 00  00 00 00 00               ........ ....
```

File: "C:\DOCUME~1\Jim\LOCALS~1\Temp\etherX... | Packets: 97 Displayed: 90 Marked: 0 Dropped: 0 | Profile: Default

Figure 13 nmap port scan

Questions

Q. Which OS is the scanned machine running?

Q. How many ports have been scanned in total? How many are open?

Using the nmap manual, or some online research:

Q. Which nmap command allows the scanning of a custom range of ports?

Q. Give an explanation of the following nmap command, and what implications it might have for an IDS?

```
nmap -T paranoid
```

Q. Why might an attacker change the timing of a scan?