Lab 5: Packet Capture & Traffic Analysis with Wireshark

Rich Macfarlane

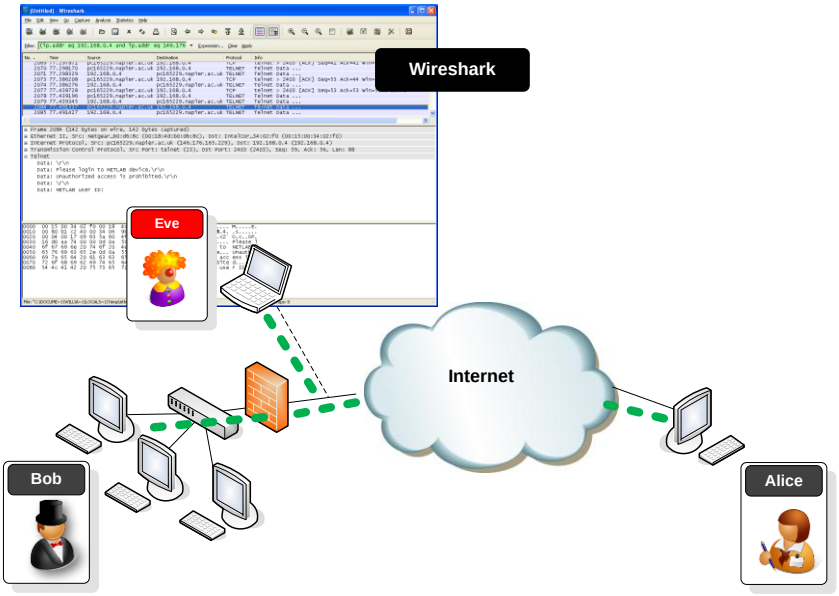
5.1 Details

Aim: This lab introduces packet capture (packet sniffing) and network traffic analysis with the **Wireshark** tool, and basic network scanning using **Nmap**.

"Chief, I think our drug-sniffing dogs need a vacation."

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.

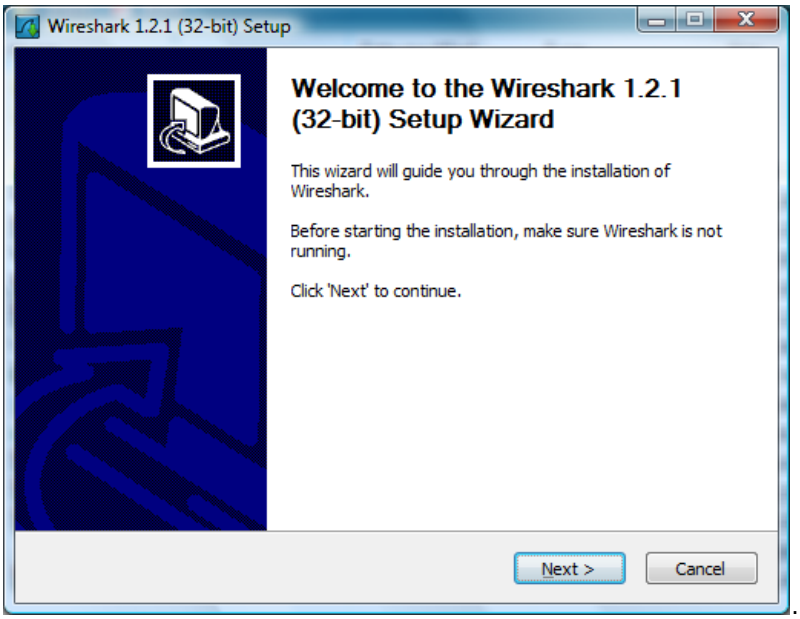
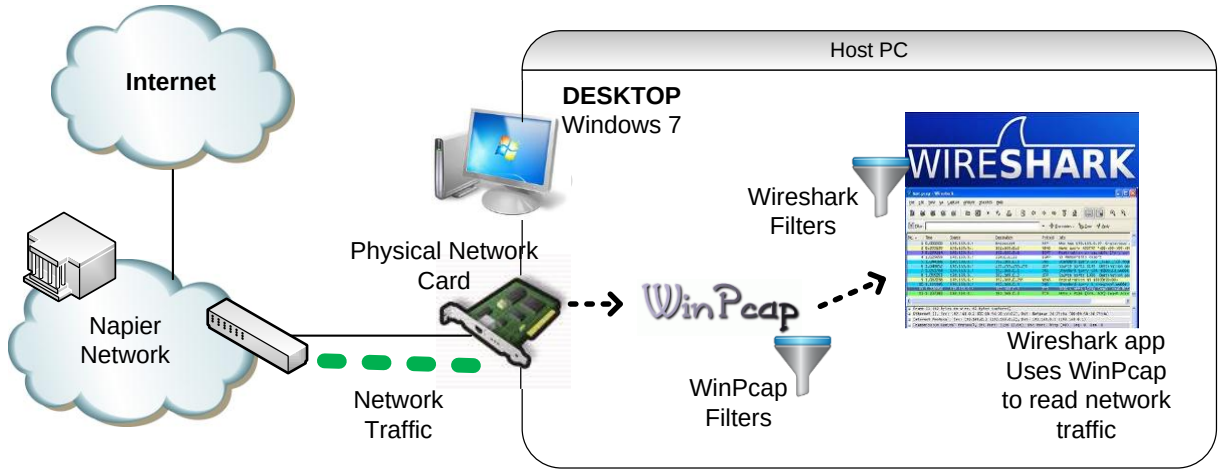


Figure 1 Wireshark Installation

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

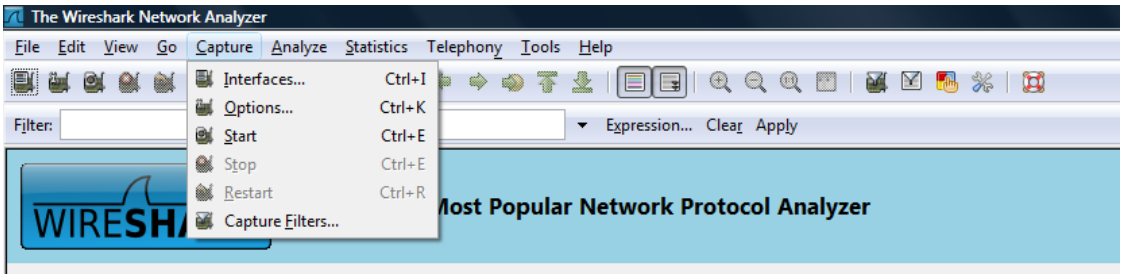
Note: On the **Install WinPcap?** window, select the install WinPcap options and select **Start WinPcap service** option, if you want to have other users besides those with administrative privileges to run Wireshark.

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.



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).

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.

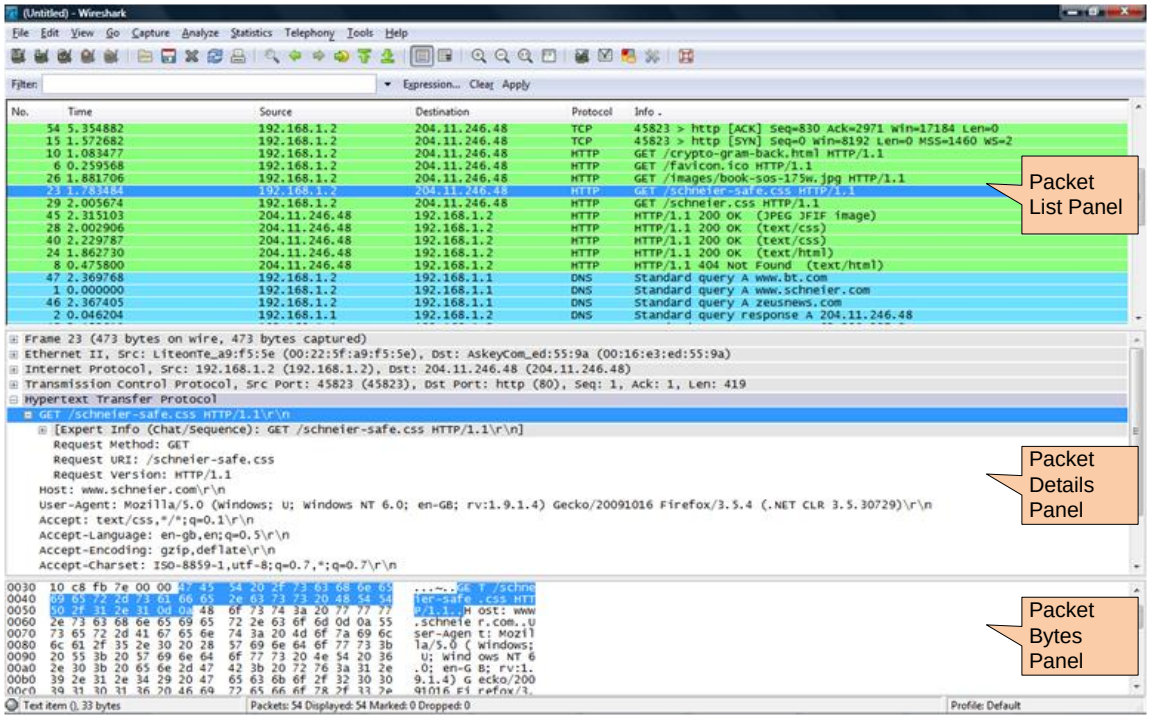


Figure 3 - Wireshark capuring traffic

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.

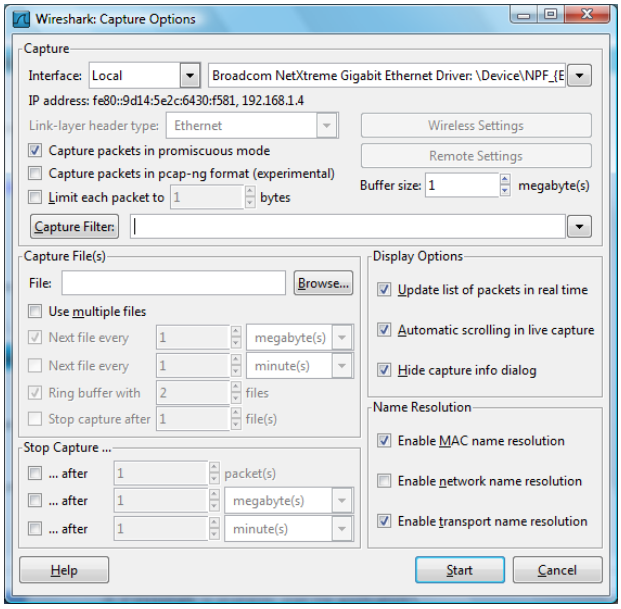


Figure 4 - Wireshark Capture Options

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**.

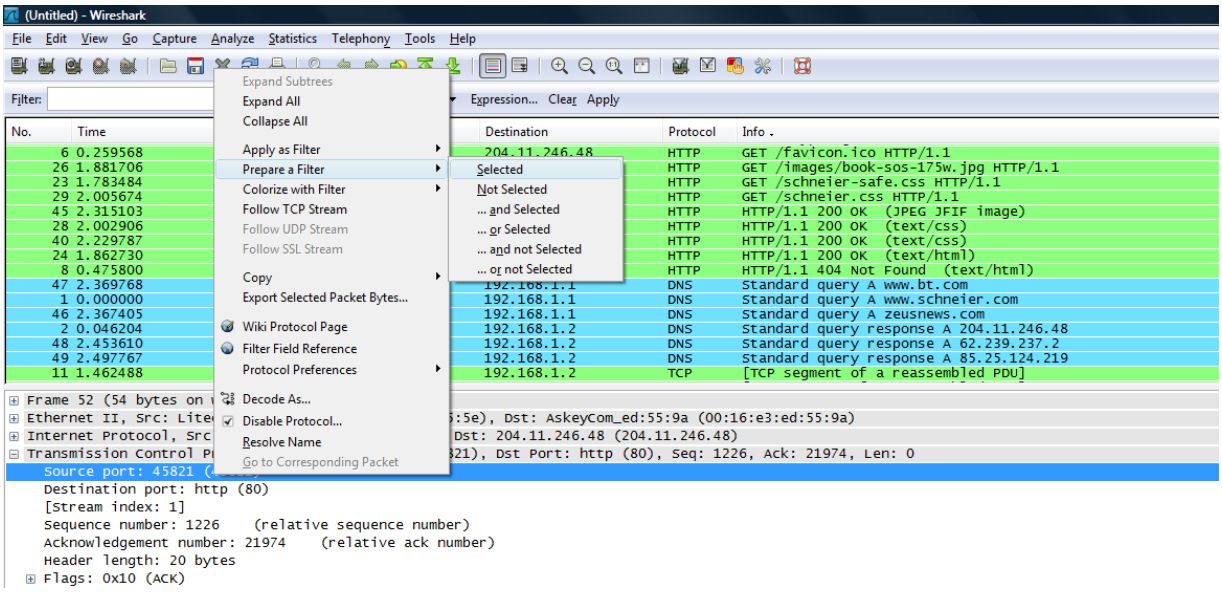


Figure 5 - Filtering on a protocol field

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.

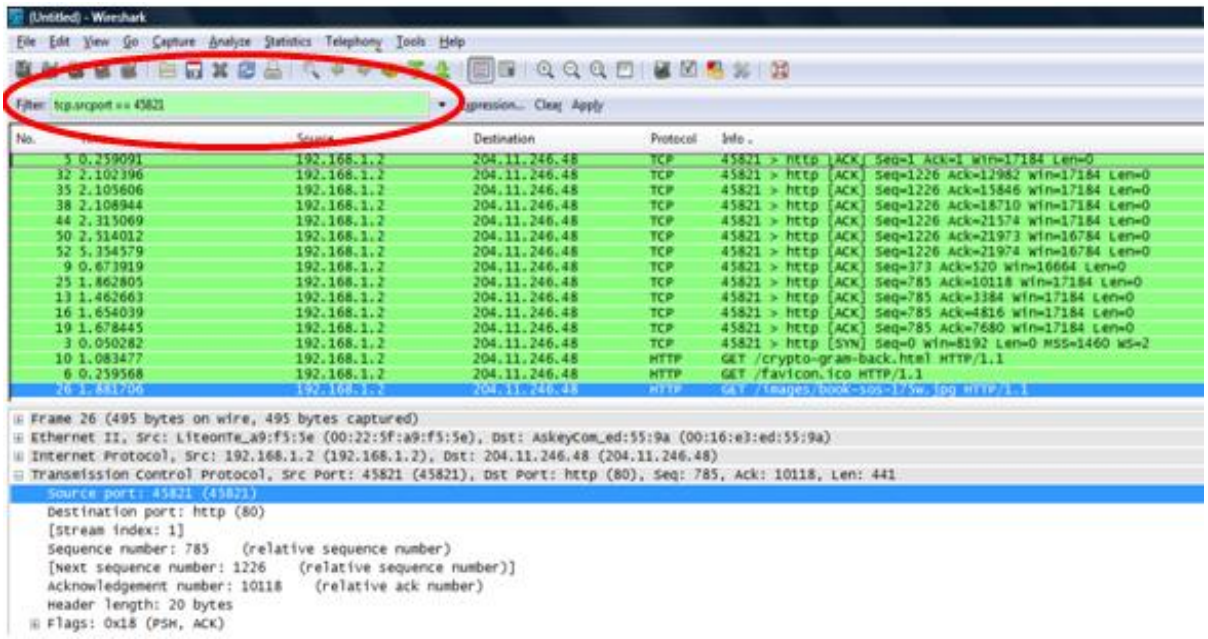
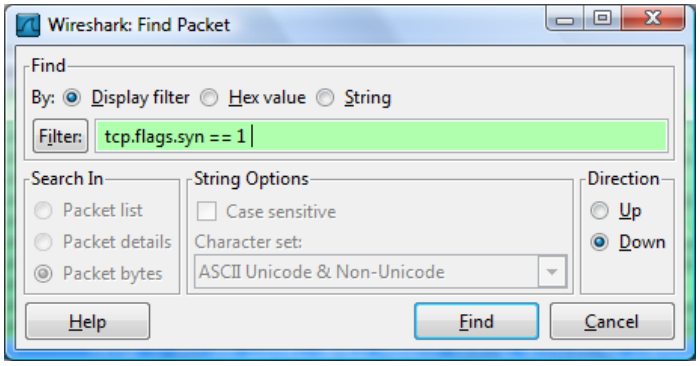


Figure 6 - Wireshark Display Filter

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.



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.

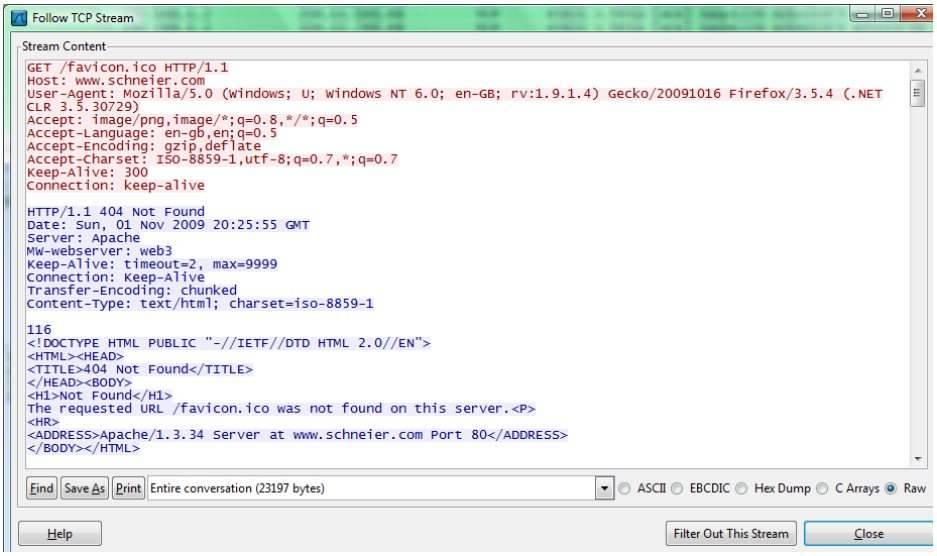
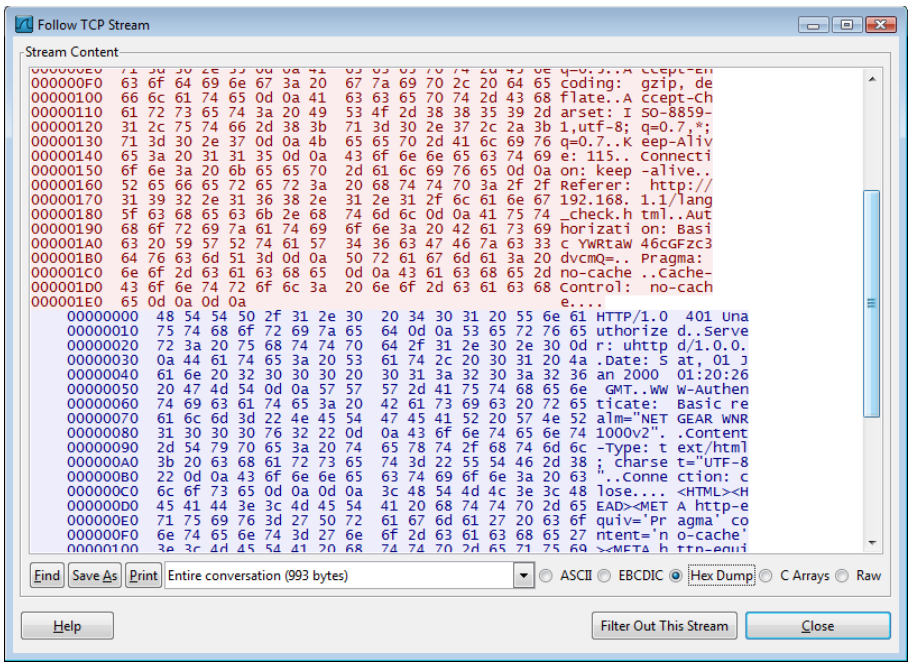
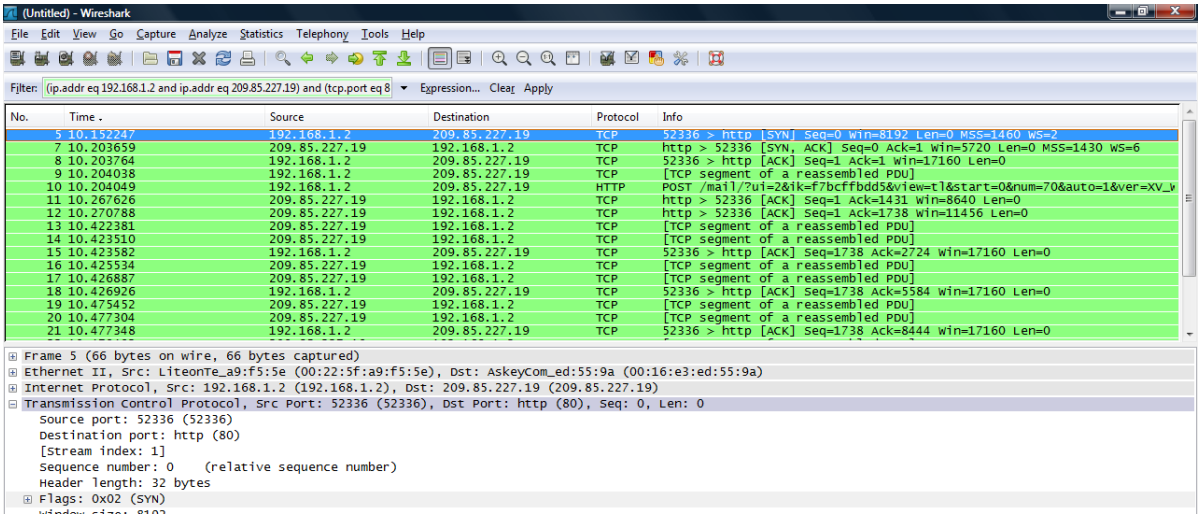


Figure 7 - Follow TCP Stream Window - ASCII

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



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.



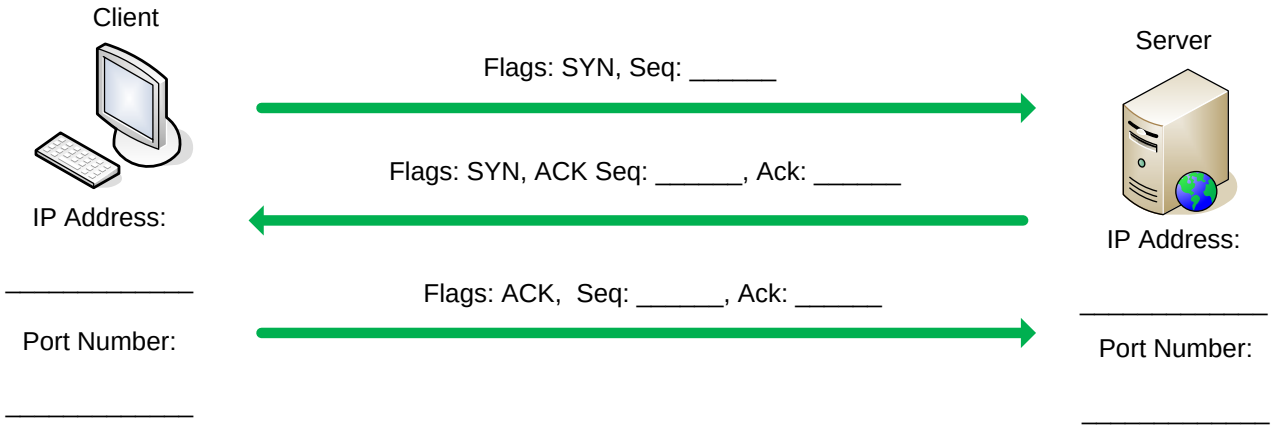
Note: Wireshark has automatically created a **display filter** to filter out this TCP conversation. In this case:

(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.

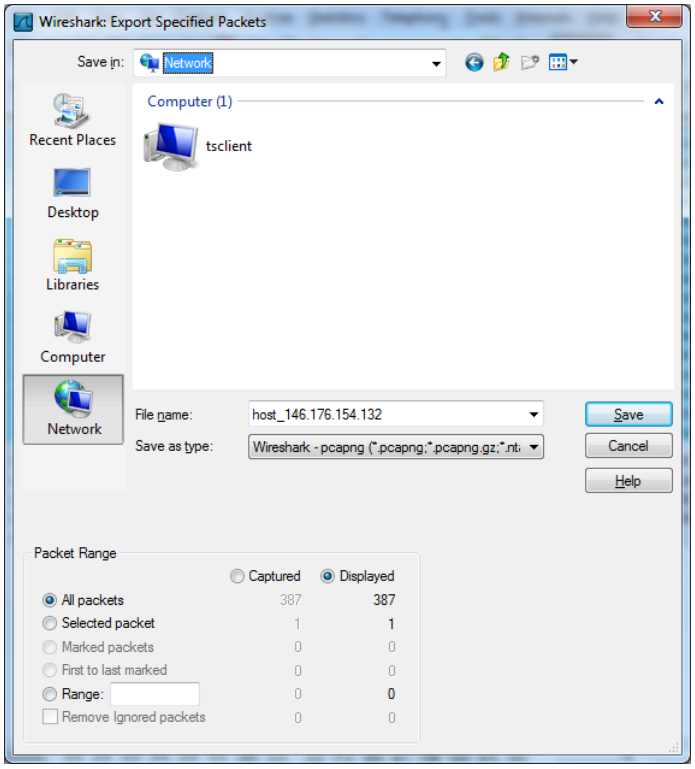
Questions

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.

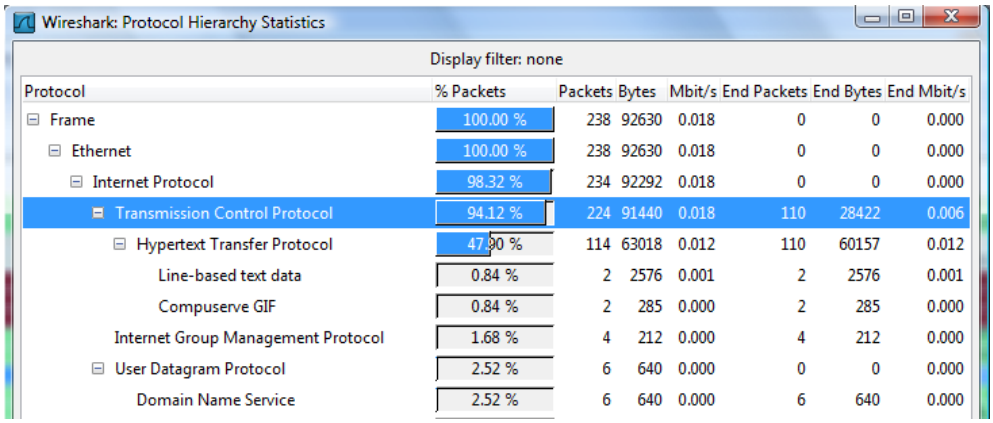


Questions

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.



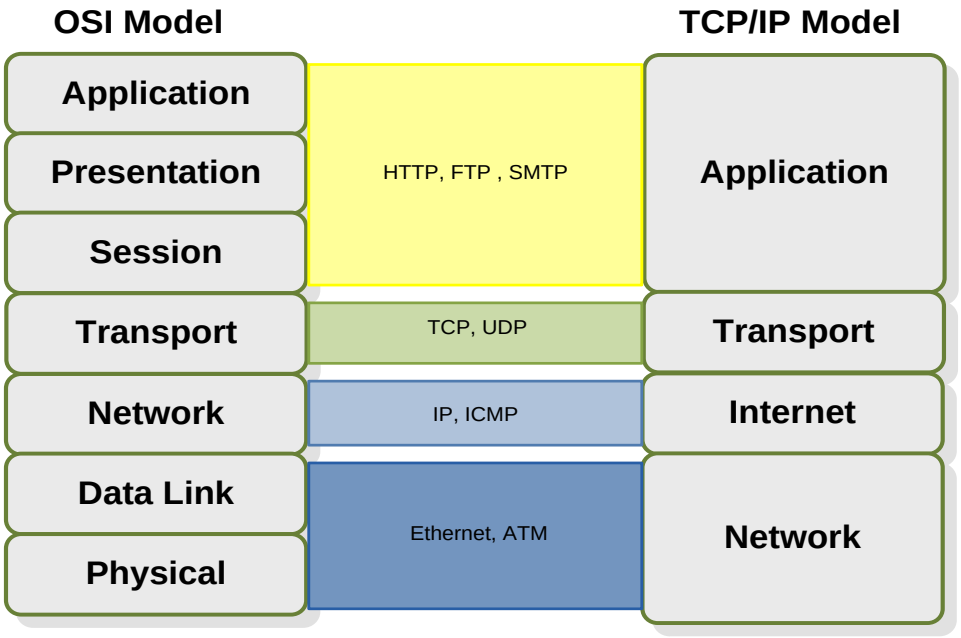
Protocol	% Packets	Packets	Bytes	Mbit/s	End Packets	End Bytes	End Mbit/s
Frame	100.00 %	238	92630	0.018	0	0	0.000
Ethernet	100.00 %	238	92630	0.018	0	0	0.000
Internet Protocol	98.32 %	234	92292	0.018	0	0	0.000
Transmission Control Protocol	94.12 %	224	91440	0.018	110	28422	0.006
Hypertext Transfer Protocol	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.

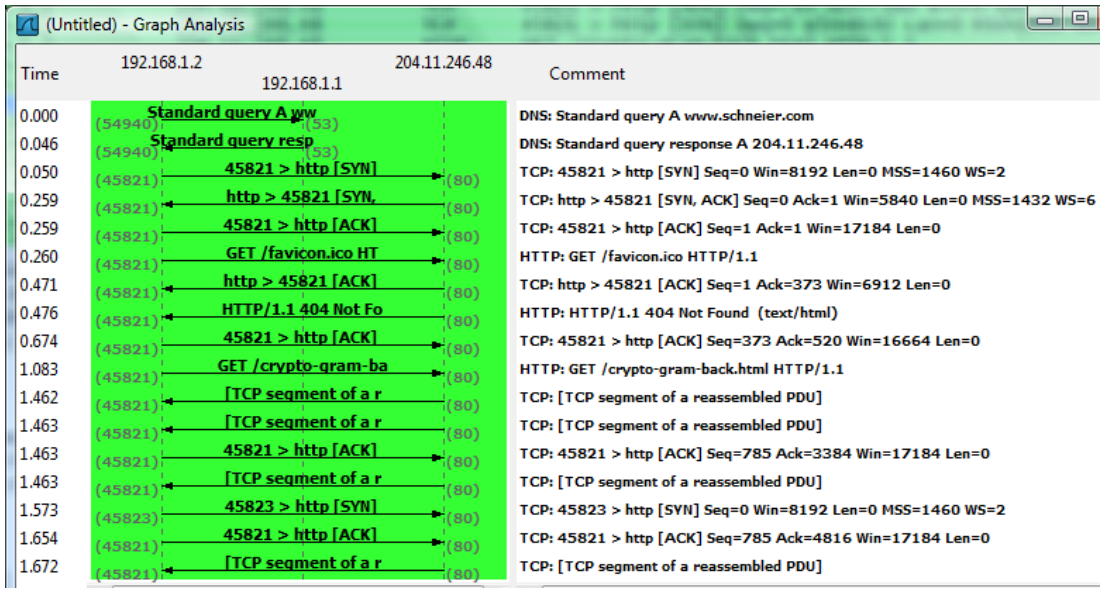
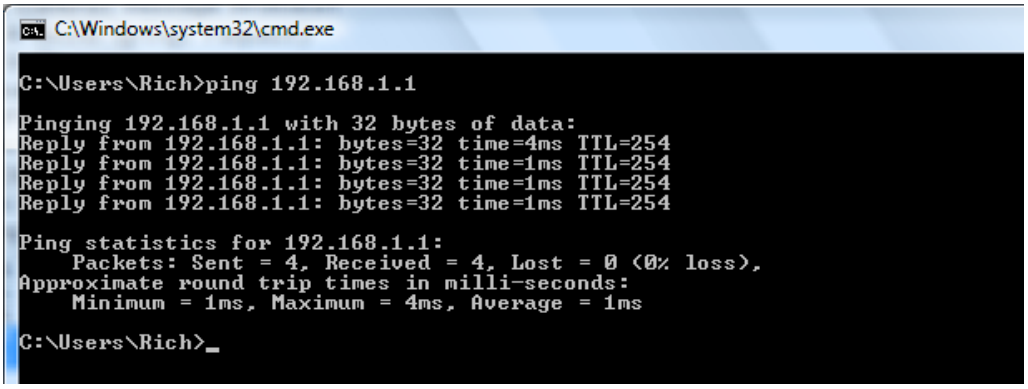


Figure 9 - Wireshark Flow Graph Window

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).



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

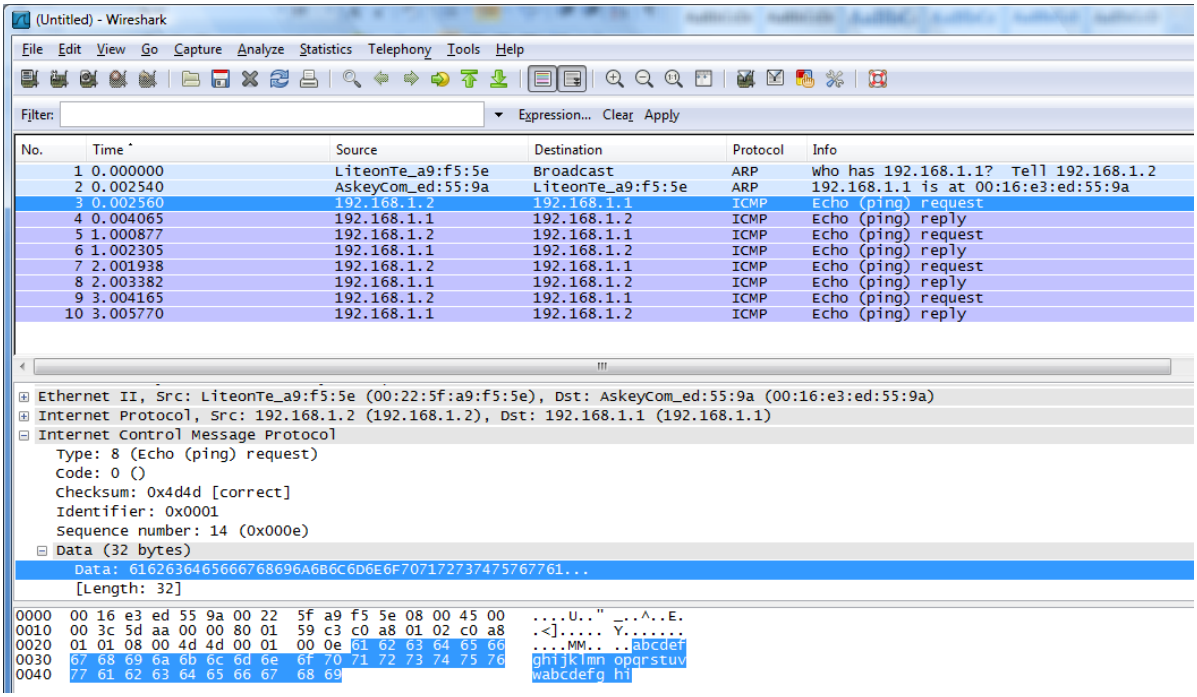


Figure 10 - Wireshark showing ICMP packets

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).

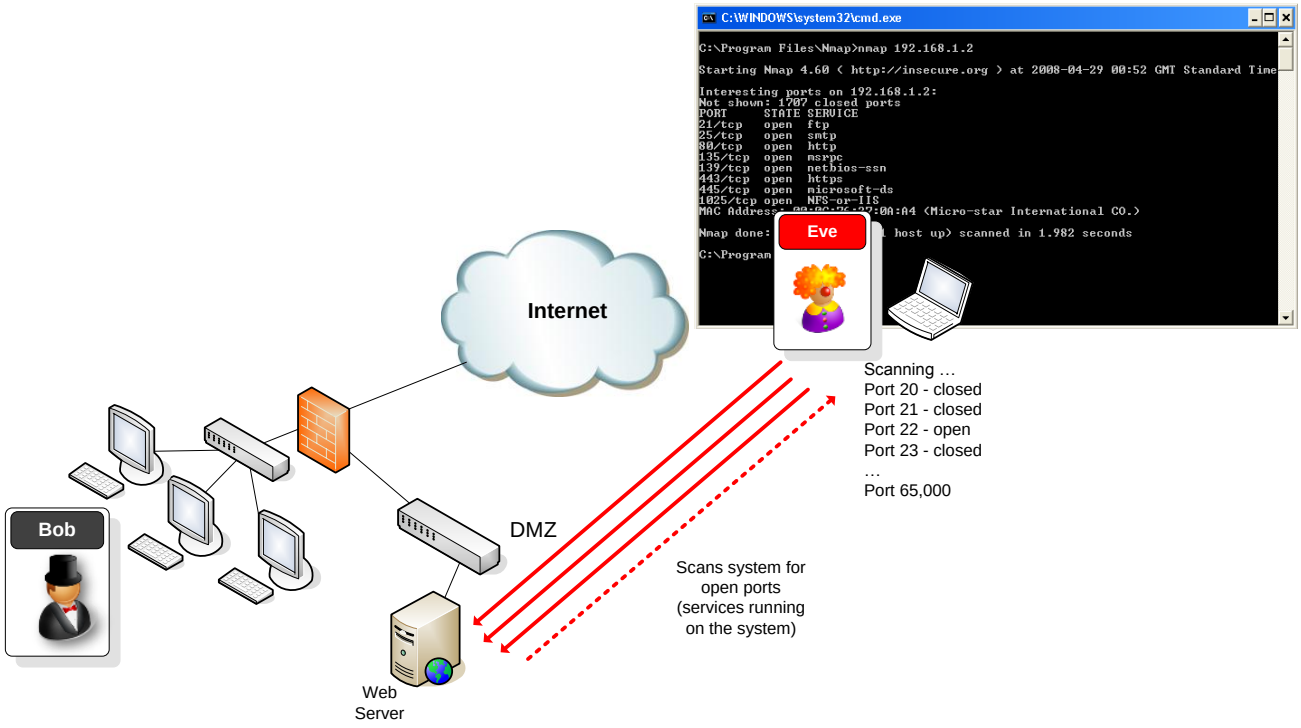


Figure 11 Port Scanning a System

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.

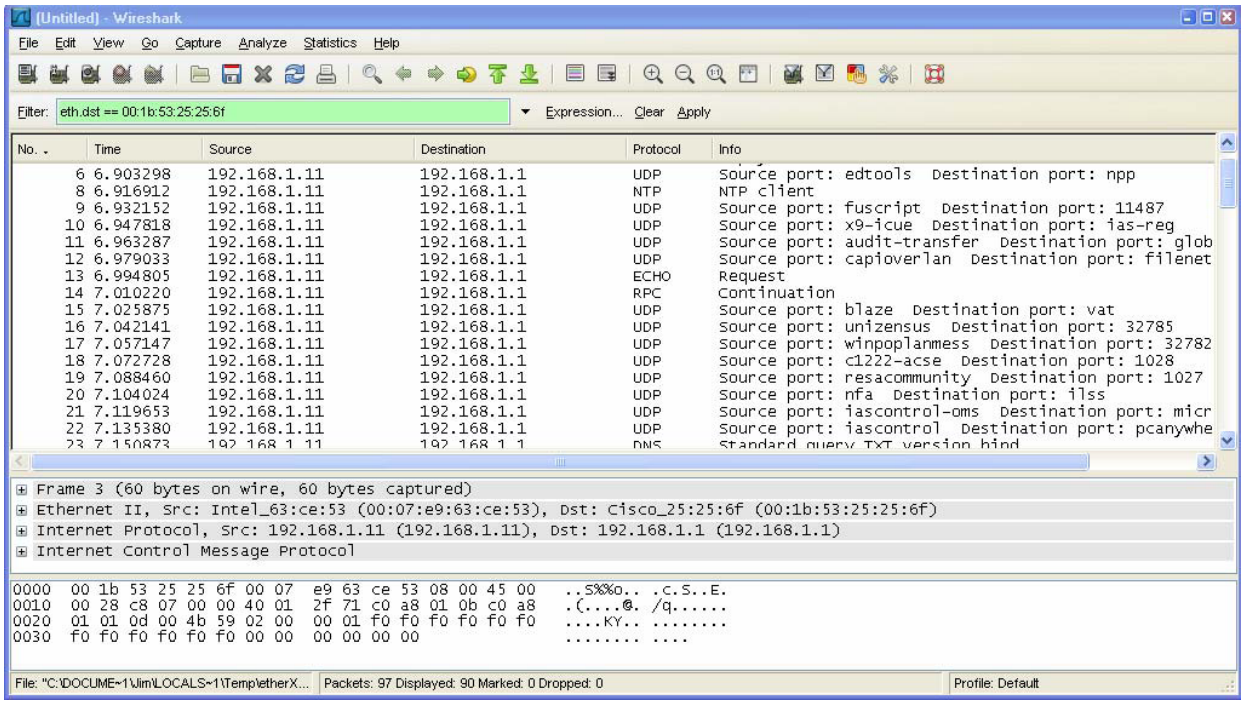


Figure 13 nmap port scan

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?