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 <u>http://www.wireshark.org/download.html</u>. 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.

📶 The Wireshark Netwo	rk Analyzer				
<u>File E</u> dit <u>V</u> iew <u>G</u> o	<u>Capture</u> <u>Analyze</u>	Statistics T	elephon <u>y</u> <u>T</u> ools	<u>H</u> elp	
	Interfaces	Ctrl+I	Þ 🗣 📦 😽		🗹 🍢 💥 🗾
F <u>i</u> lter:	iii Options iii Options iii Options	Ctrl+K Ctrl+E		✓ Expression Clear Apply	
	₩ S <u>t</u> op	Ctrl+E			
	🕷 <u>R</u> estart	Ctrl+R	Aast Dopula	Notwork Protocol Analyzor	
WIRE SH	🗃 Capture <u>F</u> ilters		nost Populai	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	S <u>t</u> op
Intel(R) 82567LM-3 Gigabit Network Connection	fe80::c1d9:9514:ccdb:43e6	91216	2	Start Options Details
进 VMware Virtual Ethernet Adapter	fe80::9de:913f:ff77:a3b4	0	0	Start Options Details
进 VMware Virtual Ethernet Adapter	fe80::bdb9:791d:cc94:96c	0	0	Start Options Details
Help				<u>C</u> lose



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					
ile Edit View Go Capture An	alyze Statistics Telephony Iools	Help			
	C C C C C C C C C C C C C C C C C C C			8 % B	
ten		Expression Clear Apply			
. 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
6 0 250568	192,108,1,2	204.11.240.48	WITP	GET /favicon ico HTTP/1.1	
26 1. 881706	192.168.1.2	204.11.246.48	HTTP	GET /images/hook-sos-175w ing HTTP/1.1	Deelvet
123 117783484	192 168 11 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	lict Danc
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 0.4/5800	204.11.246.48	192.168.1.2	HTTP	HTTP/1.1 404 NOT FOUND (Text/html)	
1 0 000000	192.100.1.2	197 168 1 1	DNS	Standard mery A www.schoolor.com	
46 2 367405	192 168 1 2	192 168 1 1	DNS	Standard query & zeusnews.com	
2 0.046204	192 168 1 1	192, 168, 1, 2	DNS	Standard query response & 204, 11, 246, 48	
ransmission control Pro opertext Transfer Proto	tocol, SFC Port: 45823 (43 Mcol	823), DST PORT: http (80), Seq: 1	, ACK: 1, Len: 419	
GET /schneler-safe.css	HTTP/1.1\r\n				
Expert Info (Chat/S)	equence): GET /schneier-sa	fe.css HTTP/1.1\r\n]			
Request Method: GET					
Request URI: /schnei	er-safe.css				Packet
Request Version: HTT	rP/1.1				
Host: www.schneier.com	n/r/n				Details
User-Agent: Mozilla/5.	0 (windows: U: windows NT	6.0: en-GB: rv:1.9.1.4) Gecko/200	91016 Firefox/3.5.4 (.NET CLR 3.5.30729)\r\n	Denel
Arcent: text/rss #/#:0	-0 1/r/n		,		Panel
Accept. Cext/cost, / iq					
Accept-Language, en-go	a /a /a /a /a				
Accept-Encouring. grip,	der face (i (i)				
Accept-charsec. 150-66	33-1, dt1-8, d=0.7, -, d=0.7 (01			
0 10 c8 fb 7e 00 00 🔐	45 54 20 2F 73 63 68 6e	S~ SE T /schne			
0 69 65 72 2d 73 61 66	65 2e 63 73 73 20 48 54	1er-safe .css HTT			Dooket
20 20 23 53 58 59 55 59	48 OF /5 /4 38 20 // //	Schoole C com II			Facket
0 73 65 72 2d 41 67 65	6e 74 3a 20 4d 6f 7a 69	ser-Agen t: Mozil			Butos
0 6c 61 2f 35 2e 30 20	28 57 69 6e 64 6f 77 73	b la/5.0 (windows;			Dytes
0 20 55 3b 20 57 69 6e	64 6f 77 73 20 4e 54 20	6 U; Wind ows NT 6			Panel
0 2e 30 3b 20 65 6e 2d	47 42 30 20 72 76 3a 31	e .0; en-G 8; rv:1.			- anor
39 31 30 31 36 20 46	69 72 65 66 6f 78 2f 33	91016 Fi refox/3.			
and items () 22 huter	Packets 54 Displayed 54 M	wheet 0 Dronneet 0		Profile: Del	ault.

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.

I 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?

I. 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 O)ptions			
Capture				
Interface: Local	 Broadcom NetXtreme Giga 	abit Ethernet Driver: \Device\NPF_{E		
IP address: fe80::9d14:5	5e2c:6430:f581, 192.168.1.4			
Link-layer header type	: Ethernet 💌	Wireless Settings		
Capture packets in	promiscuous mode	Remote Settings		
Capture packets in	pcap-ng format (experimental)	Buffer size: 1 megabyte(s)		
Limit each packet t	to 1 bytes	·····		
Capture Filter:		•		
Capture File(s)		Display Options		
File:	Browse	Update list of packets in real time		
Use <u>m</u> ultiple files				
✓ Next file every	1 megabyte(s) 🔻	Automatic scrolling in live capture		
Next file every	1 minute(s) V	✓ <u>H</u> ide capture info dialog		
Ring buffer with	2 A files	Name Perclution		
Stop capture after	1 file(s)	Name Resolution		
Stop Capture		Enable MAC name resolution		
🔲 after 1	packet(s)	Enable network name resolution		
🔲 after 1	megabyte(s) 🔻			
🔲 after 1	minute(s)	Enable transport name resolution		
Help		<u>Start</u> <u>C</u> ancel		

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**.

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

. 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.





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.

er tig-sroport == 43823		· upression Clear Apply		
-	Counts	Destination	Protocol	Into .
5 0.259091	192.168.1.2	204.11.246.48	TCP	45821 > http [ACK] Seg=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 Lerw0
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] Seg=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 W1n=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 0.239568	192.168.1.2	204.11.246.48	HTTP	GET /Tavicon.ico HTTP/1.1
26-1.881700	192.168.1.2	204.11.240.48	HALF	> GET //Images/book-sos-175w.jpg HTTP/1.1
Ethernet II, Src: Liteo Internet Protocol, Src: Transeission control Pr Source port Sizi (3) Destination port: htt	nte_a9:f5:5e (00:22:5f:a9: 192.168.1.2 (192.168.1.2) atocol. Src Port: 45821 (4 5071) p (80)	f5:5e), Dst: AskeyCom_e , Dst: 204.11.246.48 (2 5821), Dst Port: http (d:55:9a (00: 04.11.246.48 80), Seq: 78	:16:e3:ed:55:9a) 1) 15, Ack: 10118, Len: 441
[stream index: 1] Sequence number: 785 [Next sequence number Acknowledgement number	(relative sequence numb 1226 (relative sequen r:10118 (relative ack	er) ce number)] number)		

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 <u>www.schneier.com</u>, 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.

Wireshark: Find P	acket	
Find-		
By:	r 🔘 <u>H</u> ex value 🔘 <u>S</u> tring	
Filter: tcp.flags.s	yn == 1	
Search In	String Options	Direction-
Packet list	Case sensitive	© <u>U</u> р
Packet details	Character set:	Own
Packet bytes	ASCII Unicode & Non-Unicode 👻	
Help	<u> </u>	<u>C</u> ancel

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.

C Follow TCP Stream	— — X
Stream Content	
<pre>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 (CLR 3.5.30729) Accept: image/mg,image/*;q=0.8,*/*;q=0.5 Accept-Language: en-gb.en;q=0.5 Accept-Encoding: gz1p,deflate Accept-Encoding: gz1p,deflate Accept-Lindraset: ISo-8859-1.utf-8;q=0.7,*;q=0.7 Keep-Alive: 300 Connection: keep-alive</pre>	.NET E
HTTP/1.1 404 Not Found Date: sun, 01 Nov 2009 20:25:55 GMT Server: Apache MM-webserver: web3 Keep-Alive: timeout=2, max=9999 Connection: Keep-Alive Transfer=Encoding: chunked Content-Type: text/html; charset=iso-8859-1	
116 <1DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN"> <html><head> <utitle>404 Not Found </utitle></head><800Y> <h1>Not Found</h1> The requested URL /favicon.ico was not found on this server.<p> <hr/></p></html>	
<address>Apache/1.3.34 Server at www.schneier.com Port 80</address> 	-
Eind Save As Print Entire conversation (23197 bytes)	Arrays 💿 Raw
Help Filter Out This Stream	<u>C</u> lose

Figure 7 - Follow TCP Stream Window - ASCII

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

V Follow TCP Stream	
_Stream Content	
Stream Stream Content Source of 1 30 30 26 31 00 04 41 OS 05 05 10 14 20 45 06 04 05 05 05 10 14 20 45 06 04-01 31.4 Ctept=E11 Stream Content Source of 1 74 65 04 04 41 63 63 65 70 74 22 43 68 flate. A ccept=E11 00000110 61 72 73 65 74 3a 20 49 53 4f 22 38 38 35 39 22 arset: I So=8859- 00000110 31 22 75 74 66 23 83 35 71 32 02 87 72 22 3b 1,utf=8; q=0.7,*; 00000120 31 22 75 74 66 23 83 35 71 32 02 68 77 69 76 q=0.7,.K eep-alive. 00000130 71 3d 30 2e 37 0d 0a 4b 65 65 70 2d 41 6c 69 76 65 0d 0a on: keep-alive. 00000120 52 65 66 65 72 65 72 3a 20 68 74 74 70 3a 2f 2f Referer: http:// 00000120 52 65 66 65 72 65 72 3a 20 68 74 74 70 3a 2f 2f Referer: http:// 00000120 52 65 66 65 72 65 72 3a 20 68 74 74 67 36 33 2C WRTaW 46cGFzc3 00000120 58 6f 63 66 51 72 65 72 67 23 a 20 68 74 74 67 36 33 2C WRTaW 46cGFzc3 00000120 68 6f 72 69 7a 61 74 69 6 63 a 20 42 61 73 69 horizati on: Basi 00000120 64 76 62 36 51 72 65 72 67 23 a 20 66 6f 2d 63 61 63 68 65 2d no-cachecache= 00000120 64 76 62 74 72 6f 6c 3a 20 6e 6f 2d 63 61 63 68 65 2d no-cachecache= 00000120 64 77 74 68 67 67 26 97 86 54 04 0a 31 20 55 66 61 HTTP/1.0 401 Una 00000120 77 74 68 67 62 39 20	
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 Semeta h ttp-equi	-
Eind Save As Print Entire conversation (993 bytes)	Arrays 🔘 Raw
Help Filter Out This Stream	<u>C</u> lose

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					
<u>F</u> ile <u>E</u> dit	View Go Capture Analyze Stat	tistics Telephony <u>T</u> ools <u>H</u> elp)			
	4 🚳 🕷 🖻 🖬 🗶 😂 占	🔍 🗢 🔿 ዥ 👱	🔳 🖬 । 🕁 Q 🔍 🗹	M 🗹 🕴	8 % 🛱	
Filter: (ip.a	ddr eq 192.168.1.2 and ip.addr eq 209.8	5.227.19) and (tcp.port eq 8 🔻 8	Expression Clear Apply			
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=t1&start=0#=70&auto=1&ver=XV_v	
11	10.267626	209.85.227.19	192.168.1.2	TCP	http > 52336 [ACK] Seq=1 Ack=1431 Win=8640 Len=0	=
12	10.2/0/88	209.85.227.19	192.168.1.2	TCP	http > 52336 [ACK] Seq=1 ACK=1/38 Win=11456 Len=0	
13	10.422381	209.85.227.19	192.108.1.2	TCP	[ICP segment of a reassembled PDU]	
14	10.423510	209.85.227.19	192.108.1.2	TCP	[ICP segment of a reassembled PDU]	
15	10.425562	200 85 227 10	102 168 1 2	TCP	J2330 > HLLP [ACK] Seq=1/38 ACK=2/24 WHT=1/100 Len=0	
10	10.425354	209.85.227.19	102 168 1 2	TCP	[TCP segment of a reassembled PD0]	
18	10.426007	102 168 1 2	200 85 227 10	TCP	5226 S http://www.sec.instance.com	
10	10.425320	209 85 227 19	102 168 1 2	TCP	TCP segment of a reasonabled Phil	
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	-
Erame	5 (66 bytes on wire 66 by	vtes captured)				
Ether	et II Src: LiteonTe a0.f	5.5e (00.22.5f.a0.f5.5e	a) Dst: AskewCom ed:5	5·93 (00·	16:e3:ed:55:0a)	
Toton	ot Protocol Creconic_asin	1 2 (102 168 1 2)	-, 200 85 227 10 (200)	25 227 10	N	
Traper	vission Control Protocol	Fre Bort: 52226 (52226)	Det Bort: http (209.0	500.0	Jen: 0	
E II ansi		SIC FOIC: 52550 (52550)), bsc Porc. hccp (80)	, seq. 0,		
Sour	ce port: 52336 (52336)					
Dest	ination port: http (80)					
[Str	eam index: 1]					
Sequ	ence number: 0 (relativ	ve sequence number)				
Head	ler length: 32 bytes					
I Elac	IS: 0x02 (SYN)					
	low citor 9100					

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

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

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

Client	Flags: SYN, Seq:	Server
IP Address:	Flags: SYN, ACK Seq:, Ack:	// • ///
	Flags: ACK, Seq:, Ack:	IP Address:
Port Number:		Port Number:

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			- 🗿 🤌 🛤	
Ca.	Computer (1	.)			
Recent Places	tsc	lient			
Desktop					
Libraries					
Computer					
Network	File <u>n</u> ame:	host_146.	176.154.132	•	<u>S</u> ave
INCLIVITY	Save as type:	Wireshark	- pcapng (*.pcapr	ng;*.pcapng.gz;*.nt; 🔻	Cancel
					<u>H</u> elp
Packet Range		Cantured	Displayed		
Al packets		387	387		
Selected pa	icket	1	1		
Marked pace	kets	0	0		
First to last r	marked	0	0		
Range:		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 <u>www.schneier.com</u>, 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.

Wireshark: Protocol Hierarchy Statistics							
	Display filter: non	e					
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)

OSI Model		TCP/IP Model	
Application			
Presentation	HTTP, FTP , SMTP	Application	
Session			
Transport	TCP, UDP	Transport	
Network	IP, ICMP	Internet	
Data Link	Ethomot ATM	Network	
Physical	Ethemet, ATM		

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.

🗖 (Unti	itled) - Graph Analysis	11.0	
Time	192.168.1.2 192.168.1.1	204.11.246.48	Comment
0.000	(54940) (54940)		DNS: Standard query A www.schneier.com
0.046	Standard query resp (54940) (53)		DNS: Standard query response A 204.11.246.48
0.050	(45821) 45821 > http [SYN]	• (80)	TCP: 45821 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
0.259	(45821) http > 45821 [SYN,	(80)	TCP: http > 45821 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1432 WS=6
0.259	(45821) 45821 > http [ACK]		TCP: 45821 > http [ACK] Seq=1 Ack=1 Win=17184 Len=0
0.260	(45821) GET /favicon.ico HT		HTTP: GET /favicon.ico HTTP/1.1
0.471	(45821) + http > 45821 [ACK]	(80)	TCP: http > 45821 [ACK] Seq=1 Ack=373 Win=6912 Len=0
0.476	(45821) HTTP/1.1 404 Not Fo	(80)	HTTP: HTTP/1.1 404 Not Found (text/html)
0.674	(45821) 45821 > http [ACK]		TCP: 45821 > http [ACK] Seq=373 Ack=520 Win=16664 Len=0
1.083	(45821) GET /crypto-gram-b	a (80)	HTTP: GET /crypto-gram-back.html HTTP/1.1
1.462	(45821) TCP segment of a r	(80)	TCP: [TCP segment of a reassembled PDU]
1.463	(45821) TCP segment of a r	(80)	TCP: [TCP segment of a reassembled PDU]
1.463	(45821) 45821 > http [ACK]		TCP: 45821 > http [ACK] Seq=785 Ack=3384 Win=17184 Len=0
1.463	(45821) ¹⁴ [TCP segment of a r	(80)	TCP: [TCP segment of a reassembled PDU]
1.573	(45823) 45823 > http [SYN]		TCP: 45823 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
1.654	(45821) 45821 > http [ACK]	(80)	TCP: 45821 > http [ACK] Seq=785 Ack=4816 Win=17184 Len=0
1.672	(45821) TCP segment of a r	(80)	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:\Windows\system32\cmd.exe	
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.



 $\left(\int_{\Gamma_{1}} \int_{\Gamma_{2}} \int$ http://wiki.wireshark.org/DisplayFilters

🚺 (Untitled) - Wireshark						
<u>File Edit View Go Capture Anal</u>	File Edit View Go Capture Analyze Statistics Telephony Iools Help					
	Ex ex ex ex i = = = x 2 = i o, + + + + + + = = = 0, o, o, m i ex x 5, k i b;					
Filter:	▼	Expression Clear Apply				
No. Time *	Source	Destination	Protocol	Info		
1 0.000000 2 0.002540	LiteonTe_a9:f5:5e AskevCom_ed:55:9a	Broadcast LiteonTe a9:f5:5e	ARP	Who has 192.168.1.1? Tell 192.168.1.2 192.168.1.1 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		
8 2.003382	192.108.1.2	192.168.1.2	TCMP	Echo (ping) reguest		
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: LiteonT	e_a9:t5:5e (00:22:5t:a9:t5:	Se), Dst: AskeyCom_ed:5	5:9a (00:	16:e3:ed:55:9a)		
Internet Protocol, Src: 1 Thternet Control Message	92.168.1.2 (192.168.1.2), D Protocol	st: 192.168.1.1 (192.16	8.1.1)			
Type: 8 (Echo (ping) re	quest)					
Code: 0 ()						
Checksum: 0x4d4d [corre	ct]					
Tdentifier (v001						
Sequence number: 14 (0x000e)						
Data: 61626364656666768696A6866666666707172737475767761						
[Length: 32]						
0000 00 16 e3 ed 55 9a 00 22 5f a9 f5 5e 08 00 45 00 U"AE. 0010 00 3c 5d aa 00 00 80 01 59 c3 c0 a8 01 02 c0 a8 U"AE. 0020 01 01 08 00 4d 4d 00 01 00 e61 62 63 64 65 66 MMLabcdef 0030 67 68 69 6a 6b 6c 6d 6e 6f 7 07 17 27 73 74 75 76 ghijklim opgrstuw 0040 77 61 62 63 64 65 66 67 68 69						

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			
${f Q}$. After the first ping command, are the ARP and ICMP packets captured by WIre	eshark?		
	YES/NO		
I 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).



5.2.9 Using Wireshark to Capture Network Scanning Traffic

If **nmap** is not on your system, download and install from: <u>http://nmap.org/dist/nmap-5.21-setup.exe</u> 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
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 https 1025/tcp open microsoft-ds 1025/tcp open MFS-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

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**?

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

 ${f Q}$. Are the packets the same as the **ping** packets from the capture in section 6?

. What different types of packets are sent by nmap?

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 $-\circ$ 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						
Eile	Elle Edit View <u>Go</u> Capture Analyze Statistics Help					
		🖻 🖬 🗶 🎜 🛛	् 🗰 🔿 🖗 🕹		0, 🖭 🎬 🖾 畅 💥 🕽	B
<u>F</u> ilter:	Ejiter: eth.dst == 00:1b:53:25:25:6f Expression <u>Clear</u> Apply					
No. 🗸	Time	Source	Destination	Protocol	Info	
	6 6.903298 8 6.916912	192.168.1.11 192.168.1.11	192.168.1.1 192.168.1.1	UDP NTP	Source port: edtools NTP client	Destination port: npp
	9 6.932152 10 6.947818 11 6 963287	192.168.1.11 192.168.1.11 192.168.1.11	192.168.1.1 192.168.1.1 192.168.1.1		Source port: Tuscript Source port: x9-icue	Destination port: 1148/ = Destination port: ias-reg
	12 6.979033 13 6.994805	192.168.1.11 192.168.1.11 192.168.1.11	192.168.1.1 192.168.1.1 192.168.1.1	UDP ECHO	Source port: capioverl Request	an Destination port: filenet
	14 7.010220 15 7.025875	192.168.1.11 192.168.1.11	192.168.1.1 192.168.1.1	RPC UDP	Continuation Source port: blaze De	stination port: vat
	16 7.042141 17 7.057147	192.168.1.11 192.168.1.11 102.168.1.11	192.168.1.1 192.168.1.1 192.168.1.1		Source port: unizensus Source port: winpoplan	mess Destination port: 32785 mess Destination port: 32782
	19 7.088460 20 7.104024	192.168.1.11 192.168.1.11 192.168.1.11	192.168.1.1 192.168.1.1 192.168.1.1		Source port: resacommu Source port: nfa Dest	nity Destination port: 1027
	21 7.119653 22 7.135380	192.168.1.11 192.168.1.11	192.168.1.1 192.168.1.1	UDP UDP	Source port: iascontro Source port: iascontro)1-oms Destination port: micr)1 Destination port: pcanywhe
3	23 7 150873	192 168 1 11	197 168 1 1	DNS	Standard query TXT ver	sion hind
⊞ Fr	ame 3 (60 byte	s on wire, 60 byt	es captured)			
⊞ Et	hernet II, Sro	: Intel_63:ce:53	(00:07:e9:63:ce:53),	Dst: Cisco_25:2	5:6f (00:1b:53:25:25:6f)
■ Internet Plotocol, Sic. 192:100:1:11 (192:100:1:11), Sic. 192:100:1:1 (192:100:1:1) ■ Internet Control Message Protocol						
0000 0010 0020 0030	00 1b 53 25 3 00 28 c8 07 0 01 01 0d 00 4 f0 f0 f0 f0 1	25 6f 00 07 e9 63 00 00 40 01 2f 71 4b 59 02 00 00 00 fo fo 00 00 00 00	ce 53 08 00 45 00 c0 a8 01 0b c0 a8 f0 f0 f0 f0 f0 f0 00 00		E. 	
File: "C	DOCUME~1 Wim LOCA	LS~1\Temp\etherX Packe	ts: 97 Displayed: 90 Marked: 0 Dr	opped: 0		Profile: Default

Figure 13 nmap port scan

Questions

Q. Which OS is the scanned machine running?

I 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?

@ 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?