CS1840 Internet Services
Lab Session Exercise 3 - Packet Analysis

Introduction. Capturing and analysing packets from a live network is the best way to understand the format of packets that protocols send. This practical exercise focuses on packet analysis and protocol interaction from a file generated by `tcpdump`.

By default `tcpdump` dumps network packets one per line, for example (lines wrapped):

```
11:32:18.260415 192.168.1.25 > 192.168.1.10: icmp: echo request
11:32:18.260461 192.168.1.10 > 192.168.1.25: icmp: echo reply (DF)


11:33:57.659773 192.168.1.25.1227 > 192.168.1.10.80: S
   2351798711:2351798711(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
11:33:57.659832 192.168.1.10.80 > 192.168.1.25.1227: S
   679091125:679091125(0) ack 2351798712 win 8760 <nop,nop,sackOK,
   mss 1460> (DF)
11:33:57.660130 192.168.1.25.1227 > 192.168.1.10.80: . ack
   1 win 17520 (DF)
```

The first example (two lines) shows a ping from the host 192.168.1.25 to the host 192.168.1.10 and it's reply. `11:32:18.260415` is the timestamp, in `hh:mm:ss.fraction` format.

The second example shows a TCP three-way handshake between the same hosts. You'll notice the source and destination ports are tacked onto the IP addresses (e.g., 192.168.1.25.1227 where 192.168.1.25 is the IP address and 1227 is the port). Also, the SYN flag is denoted by an 'S' immediately after the colon following the destination address.port. This field can also contain 'F' for FIN, 'P' for PUSH, or 'R' for RST. If no TCP flags exist, a period ('.') is displayed. Note that the ACK flag is contained immediately after this field, since it may be set in conjunction with the other flags; whereas, the other are mutually exclusive. `2351798711:2351798711` is the byte sequence/range. The initial sequence number (ISN) is generated randomly. Since no data is exchanged at this stage, both numbers are the same. `win 16384` is the window size, or the number of bytes of buffer space the host has available for receiving data. `mss 1460` is the maximum segment size, or maximum IP datagram size that can be handled without using fragmentation. Both sides of the connection must agree on a value; if they are different, the lower value is used. `sackOK` means "selective acknowledgments", or allow the receiver to acknowledge packets out of sequence. Originally, packets could only be acknowledged in sequence. So if the third packet out of a thousand packets received went missing, the host could only acknowledge the receipt of the first two packets, and the sender would have to resend all packets from number three through one thousand. sackOK allows only the missing third packet to be re-sent. `nop`, or "no operation", is just padding. TCP options must be multiples of 4 bytes, so nop is used to pad undersized fields. `(DF)` means "don't fragment".

Please connect to teaching via the NX client to perform the practical tasks. The examples below are based on the tools available on teaching.cs.rhul.ac.uk.

1. Read information about `tcpdump` using `info tcpdump`. Look at options `-A -r -S -v -vv -X`.

2. Download the captured traffic file **sniff2.out** from the course website (Moodle). The file was generated using `tcpdump` from a local area network (Ethernet) in the department. In the file, you will find traffic related to a web session, a ftp session, a ssh session, a ping session and DNS.

3. Extract packets related to a particular application protocol, for example, `HTTP` using

   ```
   /usr/sbin/tcpdump -r sniff2.out port http
   ```

   Extract packets related to other applications

Application	Command
DNS	`/usr/sbin/tcpdump -r sniff2.out port domain`
SSH	`/usr/sbin/tcpdump -r sniff2.out port ssh`
FTP	`/usr/sbin/tcpdump -r sniff2.out port ftp`

   In general, if a command produces more output than can be displayed in your window, type the command followed by `| more` so that you can conveniently page through the output. For example `/usr/sbin/tcpdump -r sniff2.out port ftp | more` will allow you to page through all the FTP traffic.

   To store output in a file, put `> filename` after your command, as in

   ```
   /usr/sbin/tcpdump -r sniff2.out port ftp > tmpfile
   ```

   You can then view `tmpfile` at your leisure using your favorite editor.

4. Extract UDP traffic from the file. What kind of applications generate these UDP traffic?

   ```
   /usr/sbin/tcpdump -r sniff2.out udp
   ```

5. Can you find the name of a file that was transferred?

6. You are encouraged to explore further: for example,

   (a) Using the example commands above, can you identify three-way handshaking control packets and those packets used for closing the connection?

   (b) Try to decode content of packets exchanged during a ftp session. What can you observe?

   Tip: remember the options you have seen when using the info command.