Lab 12: Network Forensics & Data Hiding

12.1 Details

Aim: The aim of this lab is to investigate network traffic flows and perform analysis of them using various tools, and then look at data hiding and file signature analysis.

12.2 Activities – Network Forensics

The network forensics in first part of this lab will, involve network traffic analysis, using the modules **ProfSIMs Toolkit** and the **Wireshark Network Analyser**. Try to use both these tools, as well as opening the associated text files, and compare and contrast between.

> On-line video demo of the network capture analysis part of the lab:
> http://buchananweb.co.uk/adv_security_and_network_forensics/tcpdump01/tcpdump01.htm

Download and Install the Modules **Security & Forensic Toolkit** if you haven't already.

> The full Security Toolkit application can be downloaded from:
> http://buchananweb.co.uk/dotnetclientserver.zip

Analysis of Trace Files using ProfSIMS Toolkit

Run the **Toolkit (ProfSims)**, select the **Packet Capture->Open TCPDump** tab. This shows various **.pcap traffic capture files** created by TCPDump which we will analyse. Open the **ftp** traffic capture, using the **Open TCPDump** button, as shown in Figure 1.

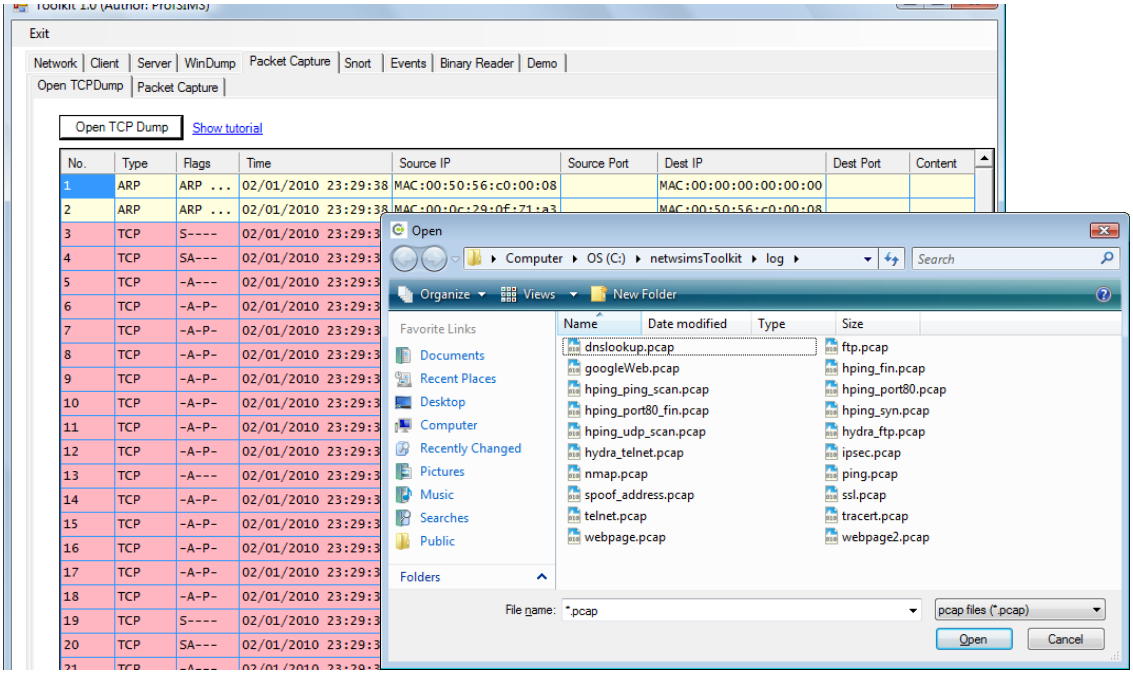
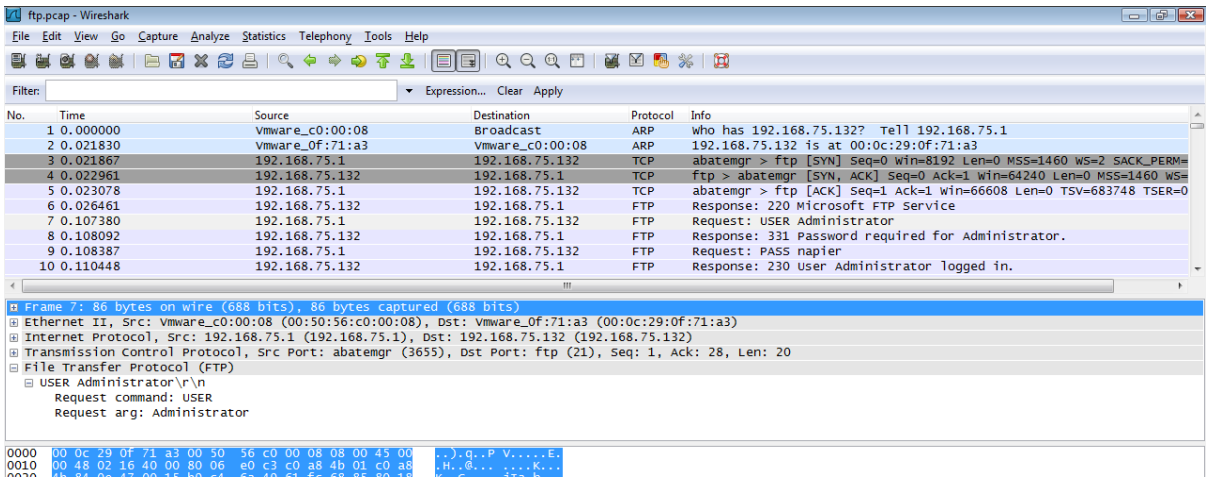


Figure 1 – Toolkit Packet Analyser, and Capture Files

Analysis of Trace Files using Wireshark

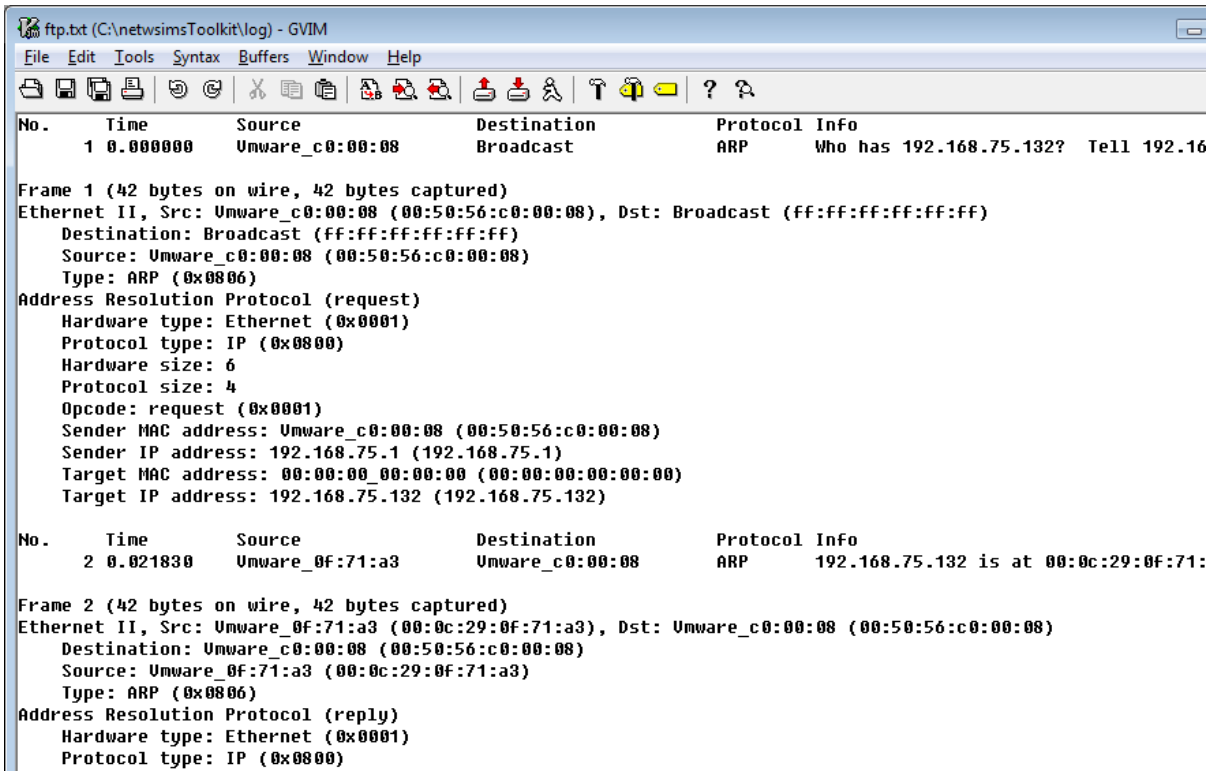
At the same time run **Wireshark**, and open the capture files from the `C:\netwsimsToolkit\log` directory, using **File>Open**.



> The captures can also be downloaded separately from the following: http://buchananweb.co.uk/log.zip

Analysis of Trace Files using Text Editor

To compare this to the raw TCPDump output (text file), also open the `C:\netwsimsToolkit\log\ftp.txt` file in a text editor, as shown below. It is very important to be able to read and understand this type of text trace, as in many situations this might be all that is available.



A solid understanding of TCP/IP Network Protocols and the TCP/IP stack implementation are essential for security and network forensic professionals. The packets are built using individual parts depending on which protocols are being used, and the fields/parameters used specific to each protocol. The TCP/IP stack, with some of the main protocols is shown below, and can be used as reference for this lab.

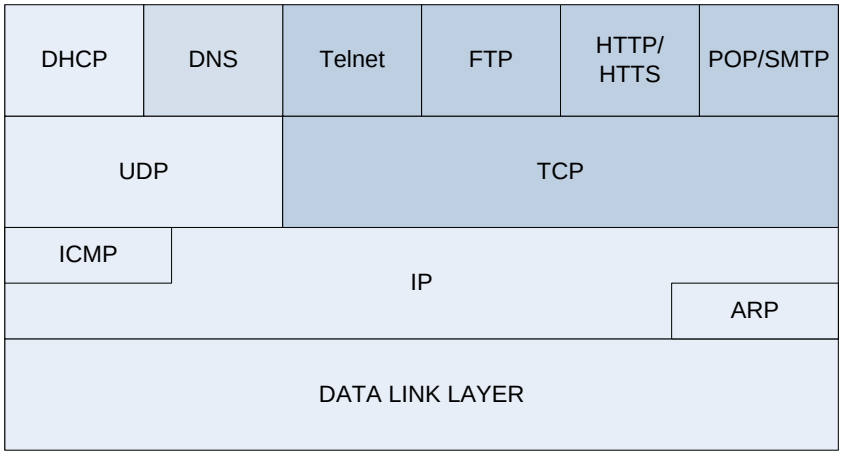
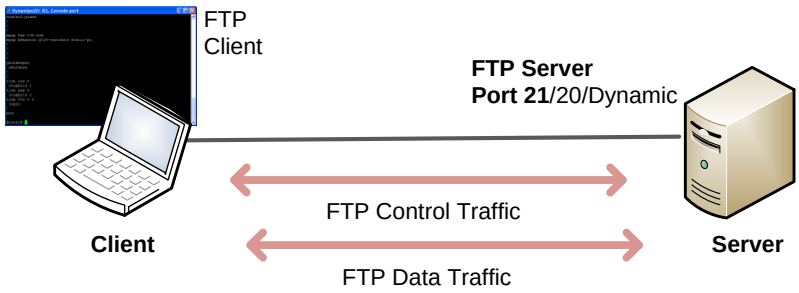


Figure 2 TCP/IP Protocol Stack

L9.1 FTP Analysis

FTP connections begin with a TCP handshake, and then clients can issue commands, and the server should respond with numerical codes. Secondary channels are used to transfer data.



Analyse the ftp traffic capture, **using the three different tools**, and determine the following:

What size are packets 1, 2, 3 and 4 (in bytes)?
1: 2: 3: 4:

Which 2 of the 3 analysis tools showed this?

What are the 3 different protocols used in the first 6 packets?

How does Ethernet header > Type field differ for these protocols?

Host src TCP port (Hint: Examine the Source Port on Packet 3):

Server src TCP port (Hint: Examine the Destination Port on Packet 3):

Host src IP address (Hint: Examine the Source IP on Packet 3):

Server src IP address (Hint: Examine the Dest IP on Packet 3):

What is the MAC address of the server (Hint: Examine the reply for Packet 2):

Identify the packets used for the SYN, SYN/ACK and ACK sequence (Hint: packets 3 to 5 look interesting):

Which users access the FTP server in the trace?
(Hint… Search the text trace, or Wireshark (Edit>Find) for the FTP client command **USER**)

Which is the return code used by the FTP server to identify:

- Password Required (Hint: Examine the content on Packet 8):

- Server type (Hint: Examine the content on Packet 12):

Which FTP command is used to determine the current working folder (Hint: Examine the content on Packet 14):

Which FTP command is used to determine the files in a folder (Hint: Examine the content on Packet 18):

Which FTP port has been used for the FTP data (hint: Examine the contents of Packets 17-19, the last two digits of the 227 response (first multiplied by 256 added to the second):

Identify the data packets used to list the contents of the FTP directory (Hint port 1046 looks interesting):

Which FTP port is being used for the FTP file transfer (hint: it is the last two digits of the 227 response (first multiplied by 256, then add the second):

Is it the same channel that was used for the FTP LIST command? YES/NO

How many Command OK response packets are in the trace?
(Hint… Search the text trace, or Wireshark for **Response: 200**)

L9.2 Telnet Analysis

Telnet connections begin with a TCP handshake, and then the server requires clients to authenticate via a login to the server, and then terminal data is communicated back and forth one character in each packet.

Open the **Telnet** traffic capture, using the ProfSIMs Toolkit, Wireshark, and a text editor, and determine the following from the telnet traffic:

Host src TCP port:

Server src TCP port:

Host src IP address:

Server src IP address:

Identify the packets used for the SYN, SYN/ACK and ACK, TCP handshake sequence:

Which packet contains the Login Banner?

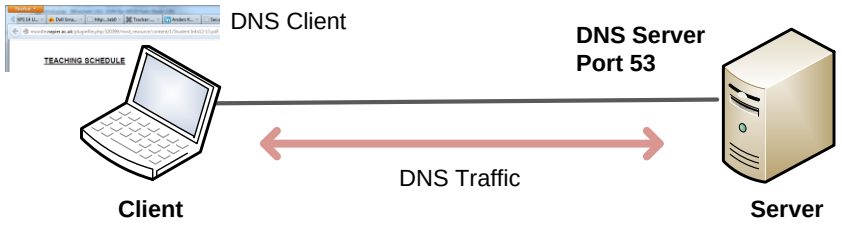
What is the login name:

What is the password:

Which commands were entered, once the Telnet connection was made?

L9.3 DNS Analysis

DNS converts Hostnames (such as www.napier.ac.uk) to IP Addresses via a query/response process. DNS typically runs over UDP, so no TCP handshake is carried out.



Open the **DNS** traffic capture, using the ProfSIMs Toolkit, Wireshark, and a text editor, and determine the following from the DNS Query/Response traffic:

What is the transport layer protocol used for DNS?

Host src UDP port:

DNS Server src UDP port:

Host src IP address:

DNS Server src IP address:

Can you identify the data packet used for **DNS A record** lookup query?

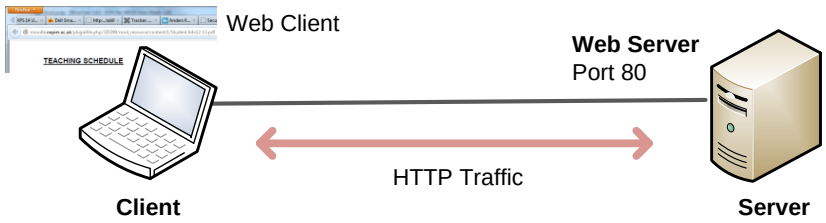
Can you identify the associated packet for DNS lookup response?

Can you find the 2 IP Addresses returned for the **www.intel.com** query?

L9.4 HTTP Web Traffic Analysis

HyperText Transfer Protocol (HTTP) is the Internet browsing protocol, and is based around plaintext messages and a query/response process. HTTP connections begin with a TCP handshake, and then Web clients such as web browsers send requests to the web server, and the server responds with HTTP status codes.

Common HTTP request commands are **GET, OPTIONS, HEAD**. Common return status codes include **200 – Success, 400 – bad request, 404 – page not found, 500 – server error**.



Open the **Web** traffic capture, using the ProfSIMs Toolkit, Wireshark, and a text editor, and determine the following from the HTTP traffic:

Host src TCP port:

Server src TCP port:

Host src IP address:

Server src IP address:

Identify the packets used for the SYN, SYN/ACK and ACK TCP handshake sequence:

What is the HTTP request command used to get the default page? (Hint: check the 1st packet after the TCP handshake)

What is the HTTP response to the successful default page request? (Hint: the content of next HTTP packet)

What is the HTTP response to the failed GET request? (Hint: the end of the trace)

Locate any GET command packet:

What is the value of the **Host** field?

List a couple of the acceptable image types listed in the **Accept** field?

L9.5 ICMP (Ping) traffic

ICMP is used for error reporting, general notifications, and can be used trouble shooting TCP/IP networks. There are several different ICMP message types. Some of the more common are **Echo**, **Redirect**, and **Destination unreachable**.

Open the **ping** traffic capture and determine the following:

Host src IP address:

Server (DNS) src IP address:

Identify the data packets used to for the ping:

How many **Echo Requests** where send from the host, and how many **Replies** where there:

Open the ICMP header for an **Echo Request**:

What is the **ICMP Type**:

Open the ICMP header for an **Echo Reply**:

What is the **ICMP Type**:

HPing Analysis

L9.6 Open **hping_fin** dump. We can see that a remote host is sending TCP segments with the FIN flag sent.

Determine the following:

Sending src TCP port range:

Receiver src TCP port:

Sending src IP address:

Receiver src IP address:

L9.7 Open **hping_port80** dump. We can see that a remote host is sending TCP segments with the SYN flag sent.

Determine the following:

Sending src TCP port range:

Receiver src TCP port:

Sending src IP address:

Receiver src IP address:

L9.8 Open **hydra_ftp** dump. We can see that a Hydra attack has been conducted on our server.

Determine the following:

Sending src TCP port range:

Receiver src TCP port:

Sending src IP address:

Receiver src IP address:

What are the logins used:

What are the passwords used:

What is the successful login/password: (the TCP stream should be unique)

L9.9 Open **hydra_telnet** dump. We can see that a Hydra attack has been conducted on our server.

Determine the following:

Sending src TCP port range:

Receiver src TCP port:

Sending src IP address:

Receiver src IP address:

What are the logins used:

What are the passwords used:

What is the successful login/password:

L9.10 Open **hping_udp_scan** dump, and determine the following:

Sending src UDP port range:

Receiver src UDP port:

Sending src IP address:

Receiver src IP address:

12.3 Activities - Data Hiding

L12.1 Run the **ProfSIMS Toolkit**, and select the **Encryption->Hashing tab.** Determine the Base-64 hash signature for the ASCII **"test"** for the following (or first few characters):

MD5:

SHA-1:

SHA-256

How many bits does each of these signatures have:

L12.2 Select **Encryption->Hash (Collision)** tab. Determine the ASCII message for the following hash signatures:

AD5F82E879A9C5D6B5B442EB37E50551

15B6AF8D85CBE1229C7150E10D5A55BD3417B40C

```
EEBC8CF2B3B360C51A34E0E8EBD98B8F37F348B7

1F7BA58706F9D405023DA32864D059C8
```

L12.3 Select **Encryption->Base-64** tab. Determine the ASCII message from the following Base-64 messages:

```
SGVsbG8gaG93IGFyZSB5b3U/

Q2FuIHlvdSByZXZlcnNlIGl0Pw==

VGhpcyBpcyBhIHNhbXBsZSBwaWVjZSBvZiB0ZXh0Li4u
```

L12.4 Select **Encryption->Private-Key Encryption** tab. Determine the Base-64 encoded strings of the ciphertext for the following strings, in 3DES and AES, which have been encrypted with the key of "sample1234":

```
napier

fullstop

apple.tree
```

How many bits does the result have, and how does it vary for the following words, and explain the reason for the changes it the output size:

```
aaaaa
aaaaaa
aaaaaaa
aaaaaaaa
aaaaaaaaa
aaaaaaaaaa
aaaaaaaaaaa
```

What does the "=" represent at the end of the encrypted string?

L12.5 Select **Encryption->Private-key Decryption** tab. The result of an encryption process is "7xCJIB1RVG5/2HQFrDH9Kw==", which was encrypted with an encryption key of either "foxtrot", "orangepeel", or "interrupt".

Which encryption key was used, what type of encryption was used, and what was the original message:

L12.6 Select **Encryption->Brute force** tab. Using a brute-force dictionary search, determine the AES encryption key for the following:

Determine the encryption key, and the original message: 2AC3B3211DEADC97C824307090BD33EA 194E22BF7A463D8A048140400497DCA7 F2BE257B9B13B72634013D9E528B6A9F 60FA30C4E4EAFF88EB741BCEEE976CD7D66DC12EBE2C9425C331F4B01FC65A2A

L12.7 Select **Encryption->Public-key encryption/decryption** tabs. Download the following public-key:

http://buchananweb.co.uk/publickey01.txt

and use it to encrypt the word "test", and prove the it can encrypt some ciphertext.

Encrypt again, is the ciphertext the same every time? **YES/NO**

L12.8 Select **Encryption->Public-key encryption/decryption** tabs. Download the following private-key:

http://buchananweb.co.uk/privatekey01.txt

and provide that it can decrypt the ciphertext.

L12.9 Select **Encryption->Public-key encryption/decryption** tabs. Using the private-key:

http://buchananweb.co.uk/privatekey02.txt

and the following cipher stream (copy it from the PDF document), determine the message:

2FB7C6F9719A05E79FA0591E92CE1884DB9CDB015F4F29D405B7ED521603AFE
B404E9884BE0F83597C3054BC721CD0F15E39091B7894B11929CACFE7B77F7A
29DD41ED3AC27D4C825157B61A1775B104045731A1B3CDD8BDDCB091544D2FA
C7D50DEBC8AD79D1BE1F73999D7FE6B8E8AB61142B71A0F274E0053D9C1FE3B
80F3

What is the message:

L12.10 Select **Encryption->Digital certificate** tab. Using the **Open CER** button, Open up digital certificate **fred.cer**, and determine its main parameters:

Certificate details:

L12.11 Select **Encryption->Digital certificate** tab. Using the **Open PTX** button, open up the password protected digital certificate: **fred.ptx** with the password apples. Use a dictionary attack to find the same password, by clicking the **Open Dictionary** button selecting the fred certificate - fred.ptx.

Open the other password protected certificate: **sample01, sample02, sample03, sample04** and **sample05**, and determine their passwords using a dictionary attack.

Passwords on certificates:

Sample01:

Sample02:

Sample03:

Sample04:

L12.12 Select **Coding->Ex-OR tab**. If the message is "Testing", what is the single digital Ex-OR key for the following Base-64 strings (**Hint**: try different single character keys):

```
NwYQFwoNBA==

EiM1Mi8oIQ==

Lh8JDhMUHQ==
```

L12.13 Select **Coding->Encoding tab**. Determine the message for the follow encoding formats:

```
48656C6C6F20686F772061726520796F753F

2431323334353637383924

VGVzdGluZyAiMTIzIiAuLi4=
```

L12.14 Select **Coding->Caesar code** tab. Determine the message for the following Caesar codes:

OLSSV OVD HYL FVB **MABL BL HGER T FXLLTZX** **PEEAT RDGT**

L12.15 Select **Binary Reader tab**. Open the first file in the drop down list: **file01**.

Refer to the Appendix A, and determine the format of the file. **What is the format of the file (such as GIF, JPEG, ZIP, etc):**

Now repeat for files 2 to 10, and complete the following table:

Name	***File format (circle correct one)***	***Is there any copyright information in the file (or associated information that is readable)?***
File2	*DOC/PPT/XLS/JPEG/GIF/WMF/ZIP*	
File3	*DOC/PPT/XLS/JPEG/GIF/WMF/ZIP*	
File4	*DOC/PPT/XLS/JPEG/GIF/WMF/ZIP*	
File5	*DOC/PPT/XLS/JPEG/GIF/WMF/ZIP*	
File6	*DOC/PPT/XLS/JPEG/GIF/WMF/ZIP*	
File7	*DOC/PPT/XLS/JPEG/GIF/WMF/ZIP*	
File8	*DOC/PPT/XLS/JPEG/GIF/WMF/ZIP*	
File9	*DOC/PPT/XLS/JPEG/GIF/WMF/ZIP*	

File10	*DOC/PPT/XLS/JPEG/GIF/WMF/ZIP*	

L12.16 Select [Binary Reader], for the ZIP file:

Identify the file name contained within the ZIP file:

What is the termination character used to terminate the file name:

Can you tell the date and time that it was last modified?

L12.17 For other binary file formats, determine their signature (if possible). **PDF** file signature:

SWF (Flash) file signature:

DLL file signature:

RTF file signature (open up a Word document, and save it in an RTF file format):

L12.18 Select [Coding], performance a frequency analysis on the following, and determine the original text:

```
XQG XP MJG PAEDM XBBKEEGQBGD XP BXC-LKMGE MGBJQXHXFT XBBKEEGO AQ MJG KDY AQ MJG
1880D. AM VYD OKG MX MJG YCGEABYQ BXQDMAMKMAXQ OGCYQOAQF MJYM Y DKERGT UG
KQOGEMYIGQ GRGET 10 TGYED. YD MJG LXLKHYMAXQ AQ MJG KDY AQBEGYDGO, AM MXXI YQ
AQBEGYDAQF YCXKQM XP MACG MX LEXOKBG MJG DMYMAD-MABD. UT MJG 1880D, AM HXXIGO
HAIGHT MJYM MJG 1880 DKERGT VXKHO QXM UG BXCLHGMG KQMAH 1890. MX XRGEBXCG MJAD,
JGECYQ JXHHGEAMJ (VJX VXEIGO PXE MJG FXRGEQCGQM) OGRADGO Y CY-BJAQG MJYM YBBGLMGO
LKQBJ BYEOD VAMJ AQPXECYMAXQ XQ MJGC. MJGDG BYEOD YHHXVGO Y BKEEGQM MX LYDD MJEXKFJ
XQHT VJGQ MJGEG VYD Y JXHG LEGDGQM.

JXHHGEAMJ'D GHGBMEXCGBJYQABYH CYBJAQG VYD GZMEGCGHT DKBBGDDPKH YQO VYD KDGO AQ MJG
1890 YQO 1900 BGQDKDGD. JG GRGQ PXKQOGO MJG BXCLYQT MJYM VXKHO HYMGE UGBXCG
AQMGEQY-MAXQYH UKDAQGDD CYBJAQGD (AUC).
```

L12.19 Select [Coding], performance a frequency analysis on the following, and determine the original text:

```
FN 1985, GLLBK TGH IGOFNE AFXXFUMBJ JFSKH. JIK HGBKH CX JIK SGUFNJCHI TKWK NCJ GH
EWKGJ GH KRLKUJKA, GNA JIK GLLBK FF TGH XGUFNE G EWKGJ AKGB CX UCSLKJFJFCN XWCS
CJIKW SGNMXGUJMWKWH. SGNY LKCLBK GJ JIK JFSK, FNUBMAFNE QFBB EGJKH, TKWK GAOFHFNE
GLLBK JC CLKN-ML JIK SGWDKJ XCW SGUFNJCHI UCSLMJKWH QY GBBCTFNE CJIKW SGNMXGUJMW-
KWH QMFBA JIKFW CTN HYHJKSH, MNAKW HJWFUJ BFUKNHK GWWGNEKSKNJH. QFBB EGJKH IGA
```

GAOFHKA JIKS JIGJ JIKY HICMBA JFK ML TFJI UCSLGNFKH HMUI GH IL GNA GJ&J. ICTKOKW, GLLBK IKBA CNJC QCJI JIKFW SGU CLKWGJFNE HYHJKS, GNA JIKFW IGWATGWK, TIFUI JIKY QKBFKOKA TKWK JCJGBBY FNJKWJTFNKA. G SGU UCMBA NCJ KRFHJ TFJICMJ QCJI FJH CLKWGJFNE HYHJKS GNA FJH IGWATGWK. WGJIKW JIGN CLKN JIK SGWDKJ ML, GLLBK AKUFAKA JC JWGSLBK UBCNKWH, KHLK-UFGBBY FN HCXJTGWK UBCNKWH. GLLBK'H XFWHJ JGWEKJ TGH AFEFJGB WKHKGWUI, TIC IGA AKOKBCLKA EKS XCW JIK LU. AFEFJGB WKHKGWUI QKBFKOKA JIGJ JIKY IGA QCWWCTKA JIK BCCD-GNA-XKKB CX JIK SGU CLKWGJFNE HYHJKS, QMJ NCJ JIK GUJMGB JKUINCBCEY. GLLBK FSSKAFGJKBY HICJ EKS CMJ CX JIK TGJKW TIKN GLLBK'H BGTYKWH, FN 1985, OFHFJKA AFEFJGB WKHKGWUI GNA JIWKGJKNKA JIKS TFJI UCMWJ GUJFCN. GJ JIK JFSK, FQS IGA QKKN DKKN JC BFUKNHK EKS XCW JIKFW CTN LWCAMUJH, QMJ JIKY TKWK XWFEIJKNKA GTGY COKW JIK XKGW CX BFJFEGJFCN, GNA JIGJ TGH JIK KNA CX EKS.

GLLBK JIKN JMWNKA JC SFUWCHCXJ JC IKGA CXX JIKFW GJJKSLJ GJ LWCAMUFNE G EMF. QFBB EGJKH, JICMEI, IGA SMUI EWKGJKW HJWKNEJI JIGN AFEFJGB WKHKGWUI GEGFNHJ GLLBK. IFH SGFN LCFNJ TGH JIGJ JIK JWMK CWFEFNGJCW CX JIK EMF TGH RKWCR. JIMH, XCW FJH SFUWCHCXJ TFNACTH, FJ TGH RKWCR'H FAKGH JIGJ TKWK QKFNE MHKA, GNA NCJ GLLBK'H. QFBB EGJKH, JICMEI, IGA GNCJIKW JWMSL UGWA: FX GLLBK TKWK ECFNE JC HJCL SFUWCHCXJ XWCS LWCAMUFNE TFNACTH JIKN SFUWCHCXJ TCMBA HJCL LWCAMUFNE GLLBFUGJFCN HCXJTGWK XCW JIK SGUFNJCHI. GLLBK DNKT JIGJ JIKY NKKAKA SFUWCHCXJ SCWK JIGN SFUWCHCXJ NKKAKA GLLBK. FN JIK XGUK CX G BGUD CX FNOKHJSKNJ FN JIKFW GLLBFUGJFCN HCXJTGWK, GLLBK HFENKA G UCNJGUJ TFJI SFUWCHCXJ TIFUI HJGJKA JIGJ SFUWCHCXJ TCMBA:

'IGOK G NCN-KRUBMHFOK, TCWBATFAK, WCYGBJY-XWKK, LKWLKJMGB, NCNJWGNHXKWGQBK BFUKNHK JC MHK AKWFOGJK TCWDH FN LWKHKNJ GNA XMJMWK HCXJTGWK LWCEWGSH, GNA JC BFUKNHK JIKS JC GNA JIWCMEI JIFWA LGWJFKH XCW MHK FN JIKFW HCXJTGWK LWCEWGSH'

TIFUI QGHFUGBBY EGOK SFUWCHCXJ UGWJK QBGNUIK XCW GBB XM-JMWK OKWHFCNH CX JIKFW HCXJTGWK, GNA TKWK PMFJK XWKK JC QCWWCT TIFUI KOKW XKGJMWKH JIKY TGNJKA. ZCIN HUMBBY GJ GLLBK HFENKA FJ, GNA EGOK GTGY CNK CX JIK SCHJ BMUWGJFOK SGWDKJH FN IFHJCWY. QGHFUGBBY, GLLBK TGH QMYFNE LKGUK TFJI SFUWCHCXJ, QMJ FJ TGH LKGUK TFJI G BCNE-JKWS UCHJ.

12.4 Toolkit (Hiding and Revealing)

On-line demo of the toolkit development part of the lab: http://buchananweb.co.uk/adv_security_and_network_forensics/toolkit05/toolkit05.htm

The objective of this series of labs is to build an integrated toolkit. Open up:

http://buchananweb.co.uk/toolkit.zip

and extract to a local folder. Next open up C# solution file **toolkit.sln**, and double click on **client.cs**.

L12.20 Select the **Binary Reader** tab. Double click on any button, and find the **get_file()** method and add the following code:

```
            try {

            byte[] buff = getBytes(fileName);


            for (int i = 0; i < buff.Length / 16; i++)
            {
                string[] arr = new string[17];
                string[] arr2 = new string[16];
                arr[0] = String.Format("{0:X2}", 16*i);

                for (int j = 0; j < 16; j++)
                {

                    arr[j+1] = buff[16 * i + j].ToString("X2");
                    char c = (char)buff[16 * i + j];
                    if ((char)c >= ' ' && (char)c <= 'z') arr2[j] = (char)c + "";
                    else arr2[j] = ".";
                }

                CreateMessageForStatusAppend(dgBytesView, arr);
                CreateMessageForStatusAppend(this.dgBytesView2, arr2);
            }

            } catch (Exception ex)
            {
            }
```

Test that you can read a GIF, JPEG and ZIP file into the binary reader.

L12.21 Double click on the "Identity file type" and add the following code:

```
            byte[] buff = getBytes(fileName);
            if (buff[0] == 0x50 && buff[1] == 0x4B) this.tbFileType.Text = "ZIP file";
            else if (buff[0] == 0xff && buff[1] == 0xD8) tbFileType.Text = "JPEG file";
            else if (buff[0] == 'G' && buff[1] == 'I' && buff[2] == 'F') tbFileType.Text =
"GIF file";
            else if (buff[0] == 0xd7 && buff[1] == 0xcd && buff[2] == 0xc6) tbFileType.Text =
"WMF file";
            else if (buff[0] == 0xd0 && buff[1] == 0xcf && buff[2] == 0x11 && buff[39] ==
0x00) tbFileType.Text = "Excel file";
            else if (buff[0] == 0xd0 && buff[1] == 0xcf && buff[2] == 0x11 && buff[39] ==
0x01) tbFileType.Text = "Word file";
            else tbFileType.Text = "Not known";
```

L12.22 Test the code, and add new file types that can be detected. Try to find a file signature for WMF, DOC, and XLS.

12.5 Appendix

JPEG file format:
FFD8 – start of image
length -- two bytes
identifier -- five bytes: 4A, 46, 49, 46, 00 (the ASCII code equivalent of a zero terminated "JFIF" string)
version -- two bytes: often 01, 02

ZIP file format:

00	ZIPLOCSIG	HEX 504B0304	;Local File Header Signature
04	ZIPVER DW	0000	;Version needed to extract
06	ZIPGENFLG	DW 0000	;General purpose bit flag
08	ZIPMTHD	DW 0000	;Compression method
0A	ZIPTIME	DW 0000	;Last mod file time (MS-DOS)
0C	ZIPDATE	DW 0000	;Last mod file date (MS-DOS)
0E	ZIPCRC	HEX 00000000	;CRC-32
12	ZIPSIZE	HEX 00000000	;Compressed size
16	ZIPUNCMP	HEX 00000000	;Uncompressed size
1A	ZIPFNLN	DW 0000	;Filename length
1C	ZIPXTRALN	DW 0000	;Extra field length
1E	ZIPNAME	DS ZIPFNLN	;filename

GIF file format:
The header is 6 bytes long and identifies the GIF signature and the version number of the chosen GIF specification. Its format is:

- 3 bytes with the characters 'G', 'I' and 'F'.
- 3 bytes with the version number (such as 87a or 89a). Version numbers are or-dered with two digits for the year, followed by a letter ('a', 'b', and so on).

WMF file format:
Standard header of: `d7 cd c6`

Excel file format:
Standard header of: `d0 cf 11 e0 a1 b1 1a`
Byte position 40(hex): `00`

Word file format:
Standard header of: `d0 cf 11 e0 a1 b1 1a`
Byte position 40(hex): `01`

PPT file format:
Standard header of: `d0 cf 11 e0 a1 b1 1a`
Byte position 40(hex): `01`