# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table of Contents</td>
<td>2</td>
</tr>
<tr>
<td>Introduction</td>
<td>5</td>
</tr>
<tr>
<td>Welcome to NetAnalysis</td>
<td>5</td>
</tr>
<tr>
<td>Feature List</td>
<td>6</td>
</tr>
<tr>
<td>NetAnalysis Key Features</td>
<td>6</td>
</tr>
<tr>
<td>Understanding the User Interface</td>
<td>7</td>
</tr>
<tr>
<td>Getting to know NetAnalysis</td>
<td>7</td>
</tr>
<tr>
<td>The Toolbar</td>
<td>8</td>
</tr>
<tr>
<td>Shortcut Keys</td>
<td>10</td>
</tr>
<tr>
<td>Status Bar</td>
<td>11</td>
</tr>
<tr>
<td>Configuring NetAnalysis</td>
<td>12</td>
</tr>
<tr>
<td>Setting properties</td>
<td>12</td>
</tr>
<tr>
<td>Identification of Suspect Computer Time Zone</td>
<td>15</td>
</tr>
<tr>
<td>Establishing a Time Zone Basis</td>
<td>15</td>
</tr>
<tr>
<td>Examining the Registry</td>
<td>15</td>
</tr>
<tr>
<td>Calculating Signed Integer Bias Values</td>
<td>17</td>
</tr>
<tr>
<td>ActiveTimeBias</td>
<td>20</td>
</tr>
<tr>
<td>Working out when Daylight Saving or Standard Time Commences</td>
<td>21</td>
</tr>
<tr>
<td>Microsoft Internet Explorer</td>
<td>23</td>
</tr>
<tr>
<td>History of a Web Browser</td>
<td>23</td>
</tr>
<tr>
<td>MSIE Browser Data</td>
<td>25</td>
</tr>
<tr>
<td>Forensic Analysis of MS Internet Explorer</td>
<td>25</td>
</tr>
<tr>
<td>Internet Explorer History List</td>
<td>26</td>
</tr>
<tr>
<td>A More Detailed Look</td>
<td>27</td>
</tr>
<tr>
<td>Daily INDEX.DAT Files</td>
<td>28</td>
</tr>
<tr>
<td>Weekly INDEX.DAT Files</td>
<td>29</td>
</tr>
<tr>
<td>Master INDEX.DAT File</td>
<td>29</td>
</tr>
<tr>
<td>Filtering Master, Daily &amp; Weekly</td>
<td>31</td>
</tr>
<tr>
<td>Temporary Internet Files</td>
<td>31</td>
</tr>
<tr>
<td>Registry Artefacts</td>
<td>34</td>
</tr>
<tr>
<td>Introduction to Registry</td>
<td>34</td>
</tr>
<tr>
<td>SHELL Folders</td>
<td>34</td>
</tr>
<tr>
<td>Days to Keep</td>
<td>35</td>
</tr>
<tr>
<td>Cache Size</td>
<td>35</td>
</tr>
<tr>
<td>Typed URLs</td>
<td>37</td>
</tr>
<tr>
<td>Internet Explorer Start Page</td>
<td>38</td>
</tr>
<tr>
<td>MS Internet Explorer Cookies</td>
<td>39</td>
</tr>
<tr>
<td>Introduction to Cookies</td>
<td>39</td>
</tr>
<tr>
<td>Persistent &amp; Session Cookies</td>
<td>39</td>
</tr>
<tr>
<td>Cookie Transport Security</td>
<td>39</td>
</tr>
<tr>
<td>First &amp; Third Party Cookies</td>
<td>40</td>
</tr>
<tr>
<td>How do cookies work?</td>
<td>40</td>
</tr>
<tr>
<td>Forensic Analysis of Cookie Data</td>
<td>42</td>
</tr>
<tr>
<td>Manual Decoding of Cookie Dates</td>
<td>44</td>
</tr>
<tr>
<td>Cookie Myths &amp; Defences</td>
<td>46</td>
</tr>
</tbody>
</table>
Introduction

Welcome to NetAnalysis

The forensic examination and analysis of user activity on the Internet can be the pivotal point of any case. With the increase in the use of computers by paedophiles and other criminals who commit crime on the Internet, it is vital for a forensic investigator to be able to extract this data and analyse it quickly and present the evidence in an understandable format.

This tool has been designed for the analysis of the internet history data. Some other forensic utilities only offer the ability to extract and print the data, which can be many thousands of URLs. How do you sift through all that data, identifying the all important evidence? NetAnalysis has a full selection of powerful tools which will allow you to identify the evidence quickly and easily.

How do you find internet history in unallocated space? NetAnalysis has its own History Extractor which will search and extract history records. The source of the evidence can be a physical write protected device, a write protected logical device, a flat file forensic DD image, a Paraben Replicator Image, mounted file or disk. NetAnalysis doesn't need to find the full internet history file, it can recover individual records.

Analyse all your internet history files at once. Put them in a folder and open them all at once, analysing all the data. Once the extraction is complete, save your work so you can go back to your analysis at any time!

NetAnalysis has a unique feature to quickly identify possible sites which cater for Indecent Images of Children/Child Abuse. This feature can be used as a guide. NetAnalysis can automatically filter out possible search criteria. This allows you to separate this vital evidence and present it as a separate exhibit. How can the suspect claim he/she stumbled across indecent images by accident if you have pages and pages of search criteria looking for that material. Imagine the look on the face of your suspect when you pull back hundreds of thousands of internet history records from unallocated space showing suspect sites and inappropriate search criteria!
Feature List

NetAnalysis Key Features

This software has a number of powerful features to allow quick and easy identification and reporting of evidence.

- Internet Explorer 3, 4, 5 and 6, and Mac Internet Explorer browser
- Netscape Communicator / Navigator up to 4.8 and Apple Mac Netscape bookmark
- Opera browser, Mac Safari, Mozilla, Firefox and AOL (ARL) File
- Extract history from unallocated space (MSIE & Safari)
- Rebuild cached pages to examine the temporary internet files
- Add comments to URL records which can be used to clearly evidence user activity
- Cookie viewer and decoder
- Multiple keyword searching - import, export & maintain your own keyword lists - examples supplied
- SQL query building and filtering
- Identify search engine criteria, usernames & passwords
- Identify yahoo, hotmail e-mail and other web based e-mail activity.
- Group URL records by category, URL, FTP, redirect, win-help activity, resource file activity, e-mail activity and host sites
- Can be used as a file viewer within forensic software
- Can be used as a file viewer from within windows (Send to).
- Copy all the history files from your forensic case and open them all together for analysis - saves time
- Save your analysis to return at a later date
- Full, fast search/filter facility to find key words within URLs, page titles, usernames and dates
- Sort and group URL records by date/time, type, content, user, page title, tag, history version etc.,
- Export to tab delimited file, CSV, Microsoft access database, PDF or HTML report
- Shows the user / profile name when URL visited
- Shows file offset and the name of the history file to identify location of URL record
- Powerful analytical printed reports
- Visit a URL by a simple mouse click
- Full tag facility to tag URL records for filtering, printing or exporting
- Copy URLs or full record to the clipboard
- Auto filtering of search criteria identifying what user has been searching for
- Auto filtering of HTTP/File/FTP/E-mail activity including identifying JPEG, movie clips and archive files
- Search and filter hundreds of thousands of URL records in a fraction of a second
- Powerful analysis of internet history without having to restore the suspect hard drive
- Extraction of history from mounted forensic images
- Analytical reports including site frequency count and summary
- Technical support forum for licensed users
Understanding the User Interface

Getting to know NetAnalysis

To get the most from the software it is important to understand the user interface and know what each feature does. In this chapter, we will look at the various components of the user interface and understand how they work.

Figure 1

NetAnalysis is rich in features and has many functions available to assist with the way the data is presented to you.
The Toolbar

Open History Files – this allows individual or multiple history files to be selected for analysis. Remember you cannot open the history files for the machine you are currently logged into, as windows will hide and prevent you from accessing them. If you wish to examine the workstation on which the software is installed, you must open the history by selecting Open All History from Folder located on the File menu.

Open Workspace File – this allows previously saved NetAnalysis Workspace files to be opened for review.

Save Workspace – this allows you to save the currently loaded data as a NetAnalysis Workspace file. You only have to press this once to save the workspace. Subsequent changes such as URL bookmarking or comments are automatically saved to the workspace.

Export Data – this function will allow you to export the workspace file in a number of different formats. Exporting as a Microsoft Access database will save all the data. Exporting as CSV, Tab Delimited or HTML will allow you to export the currently filtered records. You can also select which fields to export by switching on and off individual columns from the column menu.

Print Selected Records – this will allow the currently filtered or selected records to be printed to a simple format report.

Print-Preview – this will allow you to preview the simple format report as above.

Filter Records – this is probably one of the most important buttons on the toolbar. This will allow you to quickly filter and identify the records in which you are interested. This should be one of your main methods for filtering and finding data. This function can be quickly accessed by pressing F8.

Refresh / Clear – this function will clear any selected SQL query or filter and return you to the full recordset. This function can be quickly accessed by using F5. When the refresh button is pressed, the currently selected record will remain selected; this helps to prevent you from losing track of your position.

Column Filter Bar – this is another simple and powerful way to filter your history records. The filter bar will appear at the top of the grid, allowing you type in criteria for each field. Pressing the key below or the enter key will activate the filter. Wild cards can also be used in the query. For example, if you wanted to look at all activity between 07:00 and 07:59 hours on any particular day, you could enter 07:???:?? in the Last Visited column filter bar. You can also filter on multiple fields.
Activate Column Filter – once criterion is typed into the column filter bar, pressing this button will activate the filter.

Find First History Record – this is a quick method of finding the first Internet History URL containing a simple search word.

Find Next – used in conjunction with the Find First above, this will move the record pointer to the next URL containing the search word.

Sort Records Descending – this will perform a descending sort based on the currently selected column.

Sort Records Ascending – his will perform an ascending sort based on the currently selected column.

URL Examination Window – this function will toggle the URL examination window which will appear at the top of the grid. This window will show the full URL of the current record, allowing you to easily see all the detail.

Cookie Decoder – this button will toggle the cookie decoding window. For this to work properly, the individual cookie files must have been extracted, and their location entered in Case Properties. When a cookie record is selected – NetAnalysis will extract the cookie record and decode it for you.

Host List Window – this will toggle the host list window. Selecting this function will show each unique host in the workspace. This is a quick and easy function to examine which web sites have been visited by the suspect. If you click on any host within this window, it will filter the actual history records associated with this host.

Case Properties – this function will allow you to set case specific information such as the Case or Suspect Name along with any case summary information or case reference. The cookie and cache locations are stored here too. This information must be supplied on a case per case basis and is saved within each workspace file.

NetAnalysis Properties – this function will allow you to change the look and feel of the user interface, along with setting the date/time format and default time zone information. Time zone information must be set prior to any examination or extraction taking place. Once the time zone information has been set for a particular case, this information is saved within the workspace. This prevents you adding additional history and mixing time zones. Further information on time zones and date/time data is discussed in the chapter Understanding Date and Time Information.

Exit – this button will exit the software.
Shortcut Keys

There are a number of shortcut keys available to give you easy access to some of the key NetAnalysis functions. A summary is as follows:

- **CTRL + O**  Open History File
- **CTRL + S**  Save Workspace File
- **CTRL + D**  Decode current selected URL Record
- **CTRL + R**  Activate the advanced report preview
- **SPACE**  Tag Current Record
- **ENTER**  Add Comment to Current Record (Unless column filter bar activated)
- **F5**  Clear Filtered Records
- **CTRL + F5**  Auto Size columns to data
- **F2**  Find Previous
- **F3**  Find Next URL Record
- **F4**  Open User Keyword List
- **CTRL F4**  Open SQL Query Builder
- **F6**  Activate User Keyword List Filter
- **F7**  Find First URL Record
- **F8**  Select filter Criterion
- **F9**  Filter Tagged Records
- **CTRL + F9**  Filter Commented Records
- **F11**  Tag current record
Status Bar

The status bar shows a number of important things as follows:

<table>
<thead>
<tr>
<th>TAG</th>
<th>Filter</th>
<th>Type</th>
<th>Weekly History</th>
<th>INDEX.DAT</th>
<th>INDEX.DAT</th>
<th>Offset</th>
<th>URL Records</th>
</tr>
</thead>
</table>

Figure 3

TAG
This panel will indicate when a record that has been tagged is selected.

Filter
This panel will indicate turn blue when a filter is active.

Type
NetAnalysis will indicate from which internet history file type the data was extracted. As an example, for Microsoft Internet Explorer, the data can be from a History file (Master History), a Daily History File, a Weekly History File, a Cookie File or a Cache File. It is important to know with which type you are dealing with as the Secondary Date/Time value has a different meaning for each file type.

The example shown above indicates that the URL record in view has been extracted from a WEEKLY INDEX.DAT file. The significance of the different types is explained later in this manual.

Source
When examining a large number of Internet History Files, it is important to know exactly which file contains your evidence. This panel will show which selected source file contained the Internet history record currently selected.

Offset
In the previous panel, the entry identifies which binary file the data was extracted from. This panel will indicate the file offset where the URL record was found. This, combined with the data above, allows the forensic investigator to return to the forensic case within his/her chosen forensic software and to quickly identify the exact location on the hard drive where the data is located. With this information, it is easy to identify the PHYSICAL SECTOR and SECTOR OFFSET of the URL record, allowing any subsequent forensic investigator to quickly validate the existence of the data. In this panel, the value is shown as a decimal value. The Offset column shows the same data in decimal format.

URL Records
This panel will show the number of URL records currently displayed. On initial analysis, this panel will display the number of records found within the Internet History Files. The application of filters will alter the number of records shown.
Configuring NetAnalysis

Setting properties

It is advisable to set NetAnalysis properties before you commence as this will have an effect on everything that you do during your analysis.

The properties window can be accessed from the toolbar or File >> NetAnalysis Properties.

![Property Window](image)

The first tab shows some settings you can change to alter the look and behaviour of the grid. The grid will normally show alternate colours. You can select the even row colour to make examining data easier on the eye. The use of a light colour is recommended; harsh darker colours will cause eye strain and make data difficult to read.

Scroll tracking can be switched on or off. When scroll tracking is activated, the records and columns will change as the scroll bars are moved. When the scroll tracking is switched off, they will only move and update when it is placed in position.

When partial right column is de-selected, the right most-column will extend to the edge of the current screen.

Selecting Row Dividers will place dividing lines between each record.
The second tab will give you access to the date/time formatting screen. From here you can select the preferred date/time format for your country or select a custom format. As you change the format, the blue display will show you the effects of your selection.

The third tab will give you access to the Time Zone settings screen. It is advisable to set the time zone prior to extracting and examining any data as this will affect the accuracy of your times. If you are unsure which time zone the suspect computer was set to, this is fully explained on page 15.

**If the Time Zone of the suspect computer is not identified prior to extracting and viewing any Internet History data then the date/time stamps may not be accurately represented! Please ensure you read and understand the chapter on establishing time zone settings.**
The fourth tab allows you to set the name of the investigating forensic computer analyst. This information is shown on printed reports.
Identification of Suspect Computer Time Zone

Establishing a Time Zone Basis

In a forensic examination, establishing the Time Zone is one of the first things a forensic examiner should do. If this information is not established at an early stage and taken into account, then the validity of all Date/Time values may be brought into question. This is explained further in the chapter dealing with Date/Time values on Page 101.

Some Date/Time values stored in binary files are affected by the Time Zone setting of the original suspect computer and many Forensic applications can alter the representation of these dates by the time Zone setting of the forensic workstation. NetAnalysis is not affected by the Time Zone setting of the forensic workstation and gives the user the ability to set the translation automatically, prior to the extraction of any data. Time zone settings can be accessed via the File menu >> NetAnalysis Properties.

It is extremely important that the forensic computer analyst understands this fully. Coordinated Universal Time (UTC) is explained on page 102.

*If the Time Zone of the suspect computer is not identified prior to extracting and viewing any Internet History data then the date/time stamps may not be accurately represented!*

Examining the Registry

By examining the registry, it is possible to identify the Time Zone settings for the system at the point when it was seized. This is located in the SYSTEM hive, at the following location:

```
HKEY_LOCAL_MACHINE\System\Select
```

With reference to Figure 8 below, we can see that there are two different control sets listed at this location. Before we can establish Time Zone settings, we must establish which Control Set is in use. This information is stored in the **Current** control set key.

At this location you will find a DWORD value, which in this case equates to 0x00000001 (Decimal 1) or ControlSet001. This means that the current active control set on this system is ControlSet001.
Now that this current control set has been identified, we can navigate to that location in the registry and calculate the different values as stored. In this case, the Time Zone settings are stored here:

```
HKEY_LOCAL_MACHINE\System\ControlSet001\Control\TimeZoneInformation
```

Looking at Figure 9 below, you can see all the Time Zone information keys.

The keys are explained below. Please note that the bias settings are stored in minutes as a SIGNED integer. The bias is the difference, in minutes, between UTC and local time. All translations between UTC and local time are based on the following formula:

\[
UTC = LOCAL\ Time + Bias
\]

**ActiveTimeBias**

This value is the current time difference from UTC in minutes, regardless of whether daylight saving is in effect or not. It is this value that helps establish the current Time Zone settings. Using the formula above, take this value and add it to local time to get the UTC value.
Bias
This value is the normal Time difference from UTC in minutes. This value is the number of minutes that would need to be added to a local time to return it to a UTC value. This value will identify the Master Time Zone (Standard Time).

StandardBias
This value is added to the value of the Bias member to form the bias used during standard time. In most time zones the value of this member is zero.

DaylightBias
This value specifies a bias value to be used during local time translations that occur during daylight time. This value is added to the value of the Bias member to form the bias used during daylight time. In most time zones the value of this member is –60.

DaylightName
The Operating System uses this name during daylight saving months to display the current time Zone setting.

DaylightStart
Coded binary data to identify the date/time that Daylight Saving will commence in this time zone.

StandardName
The Operating System uses this name during daylight saving months to display the current time zone setting.

StandardStart
Coded binary data to identify the date/time that Standard Time will commence in this time zone.

DisableAutoDaylightTimeSet
This will only be visible if the setting to automatically adjust clock for daylight saving has been switched OFF.

Calculating Signed Integer Bias Values

Within digital systems, all data, whether they be numbers or characters are represented by strings of binary digits. A problem arises when you want to store negative numbers. Over the years, hardware designers have developed three different schemes for representing negative numbers: sign and magnitude, ones complement, and twos complement. The most common method for storing negative numbers is twos complement. With this method, the Most Significant Bit (MSB) is used to store the sign. If the MSB is set, then this represents a NEGATIVE number. This method affords natural arithmetic with no special rules. To represent a negative number in twos complement notation the process is simple:

- Decide upon the number of bits (n)
- Find the binary representation of the +ve value in n-bits
- Flip all the bits (change 1 to 0 and vice versa)
- Add 1
Figure 10 below shows the binary representation of the positive number 5.

+5: 0 0 0 0 1 0 1

Figure 10

To represent this as a negative number (using 8 bits) then the procedure above is followed. Flip the bits as shown above and add one as shown in Figure 11.

-5: 1 1 1 1 1 0 1 1

Figure 11

This method makes it extremely easy to add positive and negative numbers together – for example:

\[
\begin{align*}
-5 & : 11111011 \\
+5 & : +00000101 \\
\hline
& 00000000
\end{align*}
\]

Figure 12

It also makes it extremely easy to convert between positive and negative numbers:
If we look once again at the **ActiveTimeBias** in Figure 14, you will see a signed hexadecimal value. This can be calculated using twos complement.

![Registry Editor](image)

Figure 14

This value is stored in hexadecimal as a 32 bit value, so to work out the value it will need to be converted to binary. Ignore the fact that on this occasion, the registry editor is showing the decimal value (4294967236) next to it; this is purely because the registry editor does not realise the value has been stored as a signed integer.

The twos complement calculation is as follows:

\[ \text{0xFFFFFFC4 (Signed integer value)} \]

Convert this value to Binary:

\[ 11111111 11111111 11111111 11000100 \]

Or \[ 00111011 \]

Add one bit now to the value above

\[ 00111100 \]

And then convert that value back to decimal, remembering that we are dealing with a negative number:

\[-60 \text{ (Minus 60)}\]

**NOTE:** If the MSB had been zero, then the value would have been positive. With a positive value, just convert it directly to decimal. If using a scientific calculator and using the logical NOT operator, ensure you are dealing with DWORD (32 bits).
ActiveTimeBias

If there is a difference between the ActiveTimeBias and the Bias, this indicates that Daylight Saving is activated. The ActiveTimeBias will show us the number of minutes the computer was offset from UTC. Examination of the ActiveTimeBias shows the hexadecimal value 0xFFFFF04 (Decimal -60) is stored. The Bias shows the value Zero so this indicates that the Time Zone setting is GMT and that Daylight Saving is active. This is further confirmed by examining the DaylightName and StandardName keys.

This means that when the computer was last used, it was set to Daylight Saving Time which is commonly known as British Summer Time for the GMT Time Zone.

Another method to check whether the Time Zone settings are correct is to open a Daily INDEX.DAT file. In the Daily INDEX.DAT file, there are two dates stored, one in local time and one in UTC. This will be explained in greater detail on Page 104. Examination of these dates can show the offset from UTC.

<table>
<thead>
<tr>
<th>Type</th>
<th>Last Visited [UTC]</th>
<th>Secondary Date</th>
<th>Status</th>
<th>Hits</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>05/07/2005 11:51:40 Tue</td>
<td>05/07/2005 12:51:40 Tue</td>
<td>+1:00</td>
<td>4</td>
<td><a href="http://127.0.0.1:456">http://127.0.0.1:456</a></td>
</tr>
<tr>
<td>URL</td>
<td>05/07/2005 11:51:31 Tue</td>
<td>05/07/2005 12:51:31 Tue</td>
<td>+1:00</td>
<td>4</td>
<td><a href="http://127.0.0.1:456">http://127.0.0.1:456</a></td>
</tr>
<tr>
<td>URL</td>
<td>05/07/2005 11:51:24 Tue</td>
<td>05/07/2005 12:51:24 Tue</td>
<td>+1:00</td>
<td>4</td>
<td><a href="http://127.0.0.1:456">http://127.0.0.1:456</a></td>
</tr>
<tr>
<td>URL</td>
<td>05/07/2005 11:51:21 Tue</td>
<td>05/07/2005 12:51:21 Tue</td>
<td>+1:00</td>
<td>3</td>
<td><a href="http://127.0.0.1:456">http://127.0.0.1:456</a></td>
</tr>
<tr>
<td>URL</td>
<td>05/07/2005 11:50:50 Tue</td>
<td>05/07/2005 12:50:50 Tue</td>
<td>+1:00</td>
<td>3</td>
<td><a href="http://127.0.0.1:456">http://127.0.0.1:456</a></td>
</tr>
<tr>
<td>URL</td>
<td>05/07/2005 11:50:26 Tue</td>
<td>05/07/2005 12:50:26 Tue</td>
<td>+1:00</td>
<td>1</td>
<td><a href="http://127.0.0.1:456">http://127.0.0.1:456</a></td>
</tr>
</tbody>
</table>

With reference to Figure 15, the Last Visited time is shown in UTC and the Secondary time is recorded as an hour later. The status column shows the difference between both times and reflects a 1 hour difference. This is what you would expect to find when examining a machine from the UK, with the time zone set to UTC on the suspect system. As explained in the chapter on Date/Times within INDEX.DAT files, the DAILY history file in MSIE shows the LOCAL and UTC Last Visited times.

When examining a computer system, you should set the time zone in NetAnalysis Properties to Standard Time so that the Last Visited column will always show you the Standard Time. This is explained in depth in the chapter...
Understanding Date & Times. The time zone setting of the forensic workstation has no affect on the data.

Working out when Daylight Saving or Standard Time Commences

Looking at Figure 16 below, you can see two keys entitled DaylightStart and StandardStart. They hold encoded data showing the exact commencement date/time of Daylight Saving and Standard Time. To establish when daylight saving starts and ends, both keys will need to be decoded.

Figure 16

The data in DaylightStart is as follows:

```
00 00 03 00 05 00 01 00 00 00 00 00 00 00 00
```

The binary data is stored as 2 byte blocks as follows – counting from zero:

- **0x0000**: Bytes 0 & 1 refer to the year from a 1900 time base. This is only required if the change is year specific and will normally be zero.
- **0x0003**: Bytes 2 & 3 refer to the month. In this case March.
- **0x0005**: Bytes 4 & 5 refer to the week (starts at 1 and 5 means last). In this case the last week.
- **0x0001**: Bytes 6 & 7 refer to the Hour. In this case it is 0100 Hours.
- **0x0000**: Bytes 8 & 9 refer to the Minutes. In this case it is Zero minutes.
- **0x0000**: Bytes 10 & 11 refer to the Seconds. In this case it is Zero seconds.
- **0x0000**: Bytes 12 & 13 refer to the Milliseconds. In this case it is Zero milliseconds.
- **0x0000**: Bytes 14 & 15 refer to the actual Day of the Week (Sunday = 0). In this case it is Sunday.

The binary data in StandardStart can be decoded using the same method shown above.
For our example in Figure 16, Daylight Saving Time (DST) will start on Sunday of the Last Week in March at 0100 Hours. If we had decoded StandardStart, we would see that DST would end on Sunday of the last week of October at 0200 hours.
Microsoft Internet Explorer

History of a Web Browser

In the early 90s - the dawn of history as far as the World Wide Web is concerned - relatively few users were communicating across this global network. They used an assortment of shareware and other software for Microsoft Windows® operating system.

In 1995, Microsoft hosted an Internet Strategy Day and announced its commitment to adding Internet capabilities to all its products. In fulfilment of that announcement, Microsoft Internet Explorer arrived as both a graphical Web browser and the name for a set of technologies.

In July 1995, Microsoft released the Windows 95 operating system, which included built-in support for dial-up networking and TCP/IP (Transmission Control Protocol/Internet Protocol), key technologies for connecting to the Internet. In response to the growing public interest in the Internet, Microsoft created an add-on to the operating system called Internet Explorer 1.0. When Windows 95 with Internet Explorer debuted, the Internet became much more accessible for many more people.

Internet Explorer technology originally shipped as the Internet Jumpstart Kit in Microsoft Plus for Windows 95. Internet Explorer replaced the need for cumbersome, manual installation steps required by many of the existing shareware browsers.

1995: Internet Explorer 2.0

In November 1995, Microsoft released its first cross-platform browser, Internet Explorer 2.0, which supported both Macintosh and 32-bit Windows users.

With Internet Explorer 2.0 came a new set of fledgling Internet technologies that offered Web developers and designers the power to design secure, media-rich Web sites with tracking capabilities. Internet Explorer 2.0 technology introduced Secure Socket Layer (SSL) protocol as well as support for HTTP cookies, Virtual Reality Modelling Language (VRML), and Internet newsgroups.

1996: Internet Explorer 3.0

In August 1996, Microsoft released its completely rebuilt Internet Explorer technology, which included features that were revolutionary for the time. Designed for Windows 95, Internet Explorer 3.0 technology offered useful components that immediately appealed to users, including Internet Mail and News 1.0 and Windows Address Book. Later, Microsoft NetMeeting® and Windows Media Player were added. Now the Internet Explorer browser could display GIF and JPG files, play MIDI sound files, and play streaming audio files without the assistance of helper applications.

For Web developers, Internet Explorer 3.0 technology delivered a flexible programming model with a choice of scripting languages. Web designers also received more predictable results, thanks to Cascading Style Sheets (CSS). Moreover, Internet Explorer 3.0 was designed to allow Web developers to extend it easily at a time when Internet standards were quickly evolving.
1997: Internet Explorer 4.0

Designed for Windows 95, Windows 98, and Windows NT®, Internet Explorer 4.0 technology was a significant move forward. For Web developers, the addition of Dynamic HTML (DHTML) heralded the next step in Web design. DHTML gave Web developers more control over content and style and created opportunities that previously had been possible only with software applications.

Suddenly Web pages became much more interactive—users could expand menus with a click or drag images and objects around. The Web started to look more like the applications and games that people were accustomed to and less like a static series of pages.

With Internet Explorer 4.0, Microsoft Outlook® Express 4.0 was also installed for the first time as an upgrade to Internet Mail and News. This built-in component improved the way users sent, received, and organised their e-mail and address book.

1998: Internet Explorer 5.0

With the September 1998 release of Internet Explorer 5.0 technology, developers gained the ability to design richer Web applications. DHTML capabilities were expanded, giving Web developers more flexibility and power to create interactive Web sites.

Now personalisation became a key focus as Web applications based on DHTML emerged. Users encountered rich applications on the Web—for example, an expense report could automatically configure itself based on a user’s personalised settings. With expanded programming capabilities such as these, Internet Explorer 5.0 technologies helped introduce a new era of e-commerce.

2001: Internet Explorer 6

Internet Explorer 6 technology was released with Windows XP in 2001 as a more private, reliable, and flexible technology than previous versions. Because privacy and security had become customer priorities, Microsoft implemented tools that support Platform for Privacy Preferences (P3P), a technology under development by the World Wide Web Consortium (W3C).

From the home user simply browsing content on the Web, to the IT administrator deploying and maintaining a rich set of Windows Internet technologies, to the Web developer creating rich Web content, Internet Explorer 6 technologies provide the freedom to experience the best of the Internet.

2005: Internet Explorer 7 Beta

The latest version of the Internet Explorer has been released for technical evaluation, feedback, and testing by software and Web site developers. It is not currently available for general release.
MSIE Browser Data

Forensic Analysis of MS Internet Explorer

Internet Explorer stores a wealth of information regarding the activity of a user whilst connected to the internet. Finding and analysing that information can be the key to a successful forensic examination.

Internet Explorer stores web history, cookie and cache information in a binary file called INDEX.DAT. It is the INDEX.DAT file that must be extracted from your forensic image to analyse browser history and activity.

Figure 17

Internet Explorer stores web history, cookie and cache information in a binary file called INDEX.DAT. It is the INDEX.DAT file that must be extracted from your forensic image to analyse browser history and activity.

Location of History / Cookie and Temporary Internet files

The location of the INDEX.DAT file and browser data depends upon the version of Windows and whether or not user profiles are in use. Regardless of Windows version, in many cases you can't see or find the INDEX.DAT file using Windows Explorer. There is a small file called desktop.ini located in each directory where the INDEX.DAT file is located. This desktop.ini file forces Windows Explorer to hide the INDEX.DAT file and to show the content of Internet cache or history instead. The typical location is as follows:

Windows 95/98/Me (Without Profiles)

| Windows\Cookies |
| Windows\History\History.IE5 |
| Windows\Temporary Internet Files\Content.IE5 |
| Windows\UserData |

Windows 95/98/Me (With Profiles)

| Windows\Profiles <username>\Cookies |
| Windows\Profiles <username>\History\History.IE5 |
| Windows\Profiles <username>\Temporary Internet Files\Content.IE5 |
| Windows\Profiles <username>\UserData |
Internet History

The Internet Explorer History list makes it easy for the user to find and return to Web sites and pages they have visited in the past. Whether it was today or a few weeks ago, the History list can record every page the user visited, so it is easy to go back at a later date.

To return to the last page that has previously been visited, the user would click the Back button on the Internet Explorer toolbar.

- Click the Forward button to retrace your steps and return to pages you visited before you clicked the Back button (See Figure 18 below)
- To see one of the last nine pages you visited in this session, click the tiny black arrow to the right of the Back or Forward button, and then click the page you want from the list

[Figure 18: Internet Explorer History List]

Sometimes, the user will stumble across an interesting page that he/she wanted to return to at a later date. On occasion, they may have forgotten the URL or how they stumbled across it in the first place. This is when the History files are utilised.

Internet Explorer 6 automatically records Web pages visited both today and in the past. To the user, it organises them in folders on the History bar by the day you visited. Within each day, it organises the Web
sites alphabetically in folders, putting each page visited on that site in that folder. (By default, Internet Explorer stores every visit for the past 20 days).

To retrieve the web site, the user would click on the History button on the Internet Explorer toolbar (Step 1)

![History button on Internet Explorer toolbar](image1.png)

Figure 19

The user would then identify the time period they wished to search (Step 2). For example they may wish only to search through all the web sites visited on the day in question. This will then display a list of folders for each site, containing the individual pages. Step 3 would be to click on the page of interest. See Figure 20 below.

![Internet Explorer History view](image2.png)

Figure 20

In the next few pages, we will see how Internet Explorer maintains a number of History files that makes the above process possible.

**A More Detailed Look**

To examine the live internet history on a suspect machine, the data must be extracted from the forensic image and copied to a forensic workstation. The easiest way to do this is to create a local folder in which you can copy all the data. NetAnalysis was designed for forensic examination of extracted internet history and was not intended to be installed on a suspect machine. Installation of the software on a suspect machine will alter evidence and will be in breach of electronic evidence guidelines.

The Windows Operating System will try and prevent the casual user from viewing or interfering with Internet History files. For example, if you try and navigate to the data paths (as shown above) using Explorer, this view below is what you would see (Figure 21).
The first file types we will look at are the HISTORY INDEX.DAT files. There are three types of history INDEX.DAT – Master, Daily and Weekly. The Master INDEX.DAT file records the last time a particular site was visited along with a master visit count and date/time. The Daily INDEX.DAT file records the user activity for a particular day. The Weekly INDEX.DAT file is similar to the Daily INDEX.DAT file, but is created on a weekly basis by archiving the Daily INDEX.DAT files. They are stored within their own folder, the structure is shown below:

- History stored in INDEX.DAT files
- 3 History Type Files stored in different folders
  - Daily History
  - Weekly History
  - Master History

The Master INDEX.DAT file is stored in the root of the History.ie5 folder. The Daily and Weekly INDEX.DAT files are stored within their own folder as shown above. Each folder name gives a clue as to what type of INDEX.DAT file is stored.

Daily INDEX.DAT Files

The Daily INDEX.DAT file is used by Internet Explorer to record daily history and display the hierarchical information when the History button is selected. There will be a number of HOST records which are used for this process. The folder structure information is recorded as follows:
Weekly INDEX.DAT Files

Weekly INDEX.DAT files are archived daily INDEX.DAT files. At the end of a browsing week, the data from all the daily INDEX.DAT files are archived into a single weekly file. The folder structure for a weekly INDEX.DAT file is as follows:

Master INDEX.DAT File

The Master INDEX.DAT file is located in the root of History.ie5. From within Internet Explorer, the user can select a number of different views to show internet history activity.
The user can select how much history he/she wishes to keep. The default number of days for history to be kept is 20 (Hex 0x14). This can be changed in Tools >> Internet Options and can be set from 0 days to 999 days. This information is stored within a setting in the registry.

The HOST entries stored within the Daily and Weekly INDEX.DAT files are quite important to Internet Explorer. It is these entries that are used to create the more familiar history entries as shown below.

Each Visited site shown in the Internet History is recorded using a HOST entry in the INDEX.DAT File

HOST Entries are recorded in Daily & Weekly INDEX.DAT files only.
For every web site visited, it is quite possible to find multiple entries across multiple INDEX.DAT files. Internet Explorer manages the INDEX.DAT files and is constantly deleting and moving data, so there is always scope for recovering history from unallocated space. This is discussed in another chapter.

**Filtering Master, Daily & Weekly**

To view any of the history file types, select History Index Type from the Filter menu, as shown in Figure 28.

![Figure 28](image)

**Temporary Internet Files**

Temporary Internet Files (TIF) is sometimes referred to as cache. This is a location on the hard disk, where the browser can store the pictures and files which make up web pages. When the user of a web browser visits a World Wide Web page, the browser requests a copy of the page and the data which makes up the page from the web server. To allow the browser to display the page, the data is stored locally. Every time the user returns to that page, the browser will check the data it has stored locally to see if the page and its elements have been updated. Only updated pictures of pages will be requested, allowing the browser to retrieve static elements from the Temporary Internet Files location. This also serves to increase the perceived browsing speed as the browser can access locally stored pictures and pages quicker than it can download them.

The structure of the Temporary Internet Files folder is as follows:

![Figure 29](image)
The individual cache folders have randomly generated 8 digit names. The minimum number of folders for a clean installation is 4. This can rise to a great many different cache folders with extensive browsing. The size of the cache can be controlled by user settings. This is also stored in the registry and can be changed from Tools >> Internet Options.

Even if the cache is set to Zero, Internet Explorer will still take a minimum of 4Mb or 1% of logical disk – whichever is greater for storage. This is because Internet Explorer still requires a persistent cache to be able to view a page. The user cannot stop pages being cached – even by setting minimum cache values.

The default size appears to be approximately 3% of the free space of the drive. You may also find other cache folders in this location, such as the cache for Microsoft Outlook. A typical folder structure is as follows:

If you examine the content of the cache INDEX.DAT file, you will see a structure similar to the following (Figure 32). You can see the folder names at the start of the INDEX.DAT file.
Figure 32
Registry Artefacts

Introduction to Registry

The registry has a subtle but important role in Microsoft Windows operating systems. On one hand, the registry is passive – it is just a large collection of settings sitting on the hard disk, and you probably don't think much about it while you're editing a document, browsing the Internet, or searching for a file. On the other hand, it plays a key role in all those activities. The settings in the registry determine how Windows appears and how it behaves.

Windows stores configuration data in the registry. The registry is a hierarchical database, which you can describe as a central repository for configuration data (Microsoft's terminology) or a configuration database. A hierarchical database has characteristics that make it ideally suited to storing configuration data. The layout is similar to the structure of a tree or an organisational chart. The registry's hierarchical organisation makes all settings easy to reference.

SHELL Folders

One of the first keys to identify is the following (available in the NTUSER.DAT file on NT based system or USER.DAT file on 9* based systems):

HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders

This key holds the locations of all the important SHELL folders for a user. See Figure 33 below.
Internet Explorer uses the registry to store configuration settings. Careful examination of these settings, particularly when compared to the default settings can actually reveal a lot about a suspect user. For example, the default number of days to keep history is twenty. Many users reduce this value down to zero in a crude attempt to cover their browsing tracks.

If we look at Figure 33 once more, we can see the location of the Cache, Favourites and History. This is particularly important as it identifies the current location of the Internet browser data for this user. During forensic examinations, I have seen multiple history and cache folders for the same user, so it is handy to identify the current location.

### Days to Keep

This is the number of days Internet Explorer will keep history entries. The default is 20 days.

![Registry Editor](image1)

*Figure 34*

Default 0x00000014 or decimal 20.

### Cache Size

Internet Explorer needs to maintain a persistent cache to be able to render a web page. Even if cache set to Zero, it will still take a minimum of 4Mb or 1% of logical disk – whichever is greater. The user cannot prevent pages from being cached, even if the cache is set to zero. And as we have seen in other chapters, Internet Explorer is not the only application to save data to the Temporary Internet Files folder. Outlook Express uses the cache as a temporary storage location for rendering viewed email messages.

![Registry Editor](image2)

*Figure 35*
The CACHE Paths are actually stored in four separate keys. One of those keys can be seen in Figure 35. The registry key is as follows:

\texttt{HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\path1}

The formula for calculating the amount of space allocated for cache storage is as follows:

Cache Limit is \( \frac{1}{4} \) of Full Value in Kilobytes  
So \( \frac{\text{Single Path Cache Limit} \times 4}{1024} = \text{Size in Mb} \)

- Cache Limit is \( \frac{1}{4} \) of Full Value in Kilobytes  
- So \( \frac{\text{Single Path Cache Limit} \times 4}{1024} = \text{Size in Mb} \)

In Figure 35 above, the Cache Limit is set to 567169 (or hex 0x 8A781)

\( \frac{567169 \times 4}{1024} = 2215.50 \text{ Mb} \)

If we now check the Internet Explorer settings from the Control Panel, we can see our calculation is correct.
Typed URLs

Typed URLs is the name given to individual URLs that have either been typed or pasted directly into Internet Explorer address bar.

The last 25 URLs are stored in the registry so that the user can quickly navigate back to a previously visited web site. Figure 38 below shows the data stored in the registry.

The registry key is as follows:

HKCU\Software\Microsoft\Internet Explorer\TypedURLs

When a user manually clears the Internet History by pressing the Clear History button (Figure 39) located in Internet Options, this key is automatically deleted. It is recreated when a new URL is typed or pasted into the address bar.
Internet Explorer Start Page

When a user manually sets the Internet Explorer start page, the information is saved to the following location:

![Registry Editor](image)

Figure 40

The registry key is as follows:

```
HKCU\Software\Microsoft\Internet Explorer\Main\Start Page
```

There is much more information available to add to this section. This will be updated with subsequent releases of the manual.
MS Internet Explorer Cookies

Introduction to Cookies

A cookie is a text string that is included with Hypertext Transfer Protocol (HTTP) requests and responses. It is used to maintain state information as you navigate different pages on a Web site or return to the Web site at a later time.

In simple terms it is a small text file deposited on a hard drive by the browser after visiting a website that uses them. They contain certain information so that the site can recognise you on your return. Cookies store information in a format known as name/value pairs. This information can conceivably be anything from shopping cart information, username and passwords to colour preferences or localisation information.

Persistent & Session Cookies

There are two main types of cookies:

- **Persistent Cookies**
  - Written to the disk as a cookie text file
  - Secure and Non-Secure

- **Session Cookies**
  - Stored in memory and expire at the end of the session
  - Secure and Non-Secure

Cookie Transport Security

Cookies can be transported in a Secure or Non-Secure fashion. Once they are stored on the disk, the data is in plain text. However, this does not prevent developers from encrypting this data. Secure cookies can only be sent over HTTPS (SSL). Non-Secure cookies can be sent over HTTPS or regular HTTP. The title of secure is somewhat misleading. It only provides transport security.

- **Secure**
  - Secure cookies can only be sent over HTTPS (SSL).
  - SSL can only secure cookies while they are on the network
  - Once the cookie is in the browser’s end system it resides on the hard disk or memory in clear text
  - It only provides transport security

- **Non-Secure**
  - Non-Secure cookies can be sent over HTTPS (SSL) or regular HTTP
First & Third Party Cookies

The source of a cookie is of relevance to Forensic examiners. Cookies can be deposited on a suspect machine without the user realising that this has been done or without consciously visiting a specific domain. Therefore cookies can be further identified as being 1st or 3rd party cookies.

- **1st Party Cookie**
  - A first-party cookie either originates on or is sent to the Web site you are currently viewing
  - These cookies are commonly used to store information, such as your preferences when visiting that site.

- **3rd Party Cookies**
  - A third-party cookie either originates on or is sent to a Web site which is different from the one you are currently viewing
  - Third-party Web sites usually provide some content on the Web site you are viewing.
  - Many sites use advertising from third-party Web sites and those third-party Web sites may use cookies.
  - A common use for this type of cookie is to track your Web page use for advertising or other marketing purposes.
  - Third-party cookies can either be persistent or session (temporary).

How do cookies work?

To understand how cookies actually work, you need to have an understanding of how HTTP works, and the request/response cycle. The HTTP protocol is discussed at length on page 81.

HTTP is a request/response protocol between clients and servers. An HTTP client, such as a web browser, typically initiates a request by establishing a TCP connection to a particular port on a remote host (port 80 by default). An HTTP server listening on that port waits for the client to send a request string, such as "GET / HTTP/1.1" (which would request the default page of that web server), followed by an email-like MIME message which has a number of informational header strings that describe aspects of the request, followed by an optional body of arbitrary data. Some headers are optional, while others (such as Host) are required by the HTTP/1.1 protocol. Upon receiving the request string (and message, if any), the server sends back a response string, such as "200 OK", and a message of its own, the body of which is perhaps the requested file, an error message, or some other information.

When a web browser makes a request to a web server for a page, the information is sent to the web server in the format of a GET command. In this first visit to the site, there is no stored cookie and therefore no additional information to be sent. Figure 41 outlines a simple GET command as sent to retrieve the Google search page.
Once this request is made, the web server responds with the following:

```
GET / HTTP/1.1
Host: www.google.com
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/xbm, */*
Accept-Language: en
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.4322)

Figure 41
```

Once this request is made, the web server responds with the following:

```
HTTP/1.1 200 OK
Transfer-Encoding: chunked
Date: Sat, 04 Sep 2004 09:58:56 GMT

Content-Type: text/html
Cache-Control: private
Server: GWS/2.1
Set-Cookie: PREF=ID=6ct8e42301f6c6d1:TM=1094291937:
    LM=1094291937:S=1icZAD9r1YqVW.Pm;
    expires=Sun, 17-Jan-2038 19:14:07 GMT; path=/; domain=.google.com

Figure 42
```

As well as providing the main Google search page, the server uses a command **Set-Cookie** which instructs
the browser to create a cookie for this domain. As you can see from the information above, this cookie has a number of name/value pairs and also instructs the browser as to when this cookie will expire. In this particular cookie, there are multiple name/value pairs separated by a colon.

Now that the main page has been obtained, the browser will parse the data and make further requests for the additional content/resources. Everyone who is familiar with the Google search page will know that the Google logo is usually in the centre and at the top of the page. The next request will be for the Google logo.
In Figure 43, we can see a further GET command has been issued by the browser for the LOGO.GIF file. We can also see that when compared to the initial request in Figure 41, there is some other information. This additional information is the browser passing back to the web server the information which was stored in the cookie.

Sometimes it is possible to identify some of the data stored inside the cookie. In the case above, there are two date/time values sent back to the server.

```
TM=1094291937
LM=1094291937
```

These numbers are UNIX time stamps (number of seconds from 1st January 1970 00:00:00 Hours UTC) and convert to **Sat, 04 September 2004 09:58:57 UTC**.

Every time the user returns to that page or a request is made for a resource from that domain, the web browser will send the cookie information with the request. However, once the cookie expires or has been deleted, a new cookie will be set.

**Forensic Analysis of Cookie Data**

Each cookie is a simple text file stored at the following locations.

**Windows 95/98/Me (Without Profiles)**

```
\Windows\Cookies\`
```

**Windows 95/98/Me (With Profiles)**

```
\Windows\Profiles\<username>\Cookies\```
Windows NT
\Winnt\Profiles\<username>\Cookies\

Windows 2000 & XP
\Documents and Settings\<username>\Cookies\

Opening the Cookie folder within Explorer will show you the view in Figure 44. The cookie folder contains an INDEX.DAT file which acts as a database of cookie records. The cookie records are small text files which are named to reflect the host which set it. If a cookie is set by a page that is not in the root of the domain, then the cookie name will reflect that page and not the domain.

![Figure 44](image)

Each Cookie contains 6 fields which are identified as follows:

- **NAME** of the cookie
- **VALUE** of the cookie
- **MODIFICATION** date
- **EXPIRY** date
- **PATH** and **DOMAIN** for which it is valid
- **SECURITY** Tag

![Figure 45](image)
The easiest way to examine the raw data in a text cookie is to open it using WordPad. If you open a cookie in Notepad, the data will not be displayed properly because Notepad expects a carriage return and line feed (0x0D, 0x0A) after each line, where cookie lines are separated by a Line Feed (0x0A) only.

The Expiry and Modification date/time is stored as two separate numbers. This pair of values is a 64bit FILETIME and is covered more on page 101. The security tag is identified by a simple integer and can be many different values. The rule of thumb is if the security tag value is divisible by 2, then it is NOT secure.

Manual Decoding of Cookie Dates

Cookie date/times can be decoded easily within NetAnalysis, but can also be decoded individually using DCode, the free date/time decoder. An example is shown below in Figure 47.
The next chapter outlines the procedure for examining Internet History, Cache and Cookie data. Remember that once a cookie folder has been exported from your case for examination, you will need to make sure that NetAnalysis knows the location of the COOKIE folder. This may be set automatically if you have selected the option to open all history from folder. If this is not the case, the location has to be set manually by selecting Case Properties from the File menu.

![Figure 48](image)

The full path of the cookie folder can then be selected by pressing the browse button. To view and decode a cookie, make sure that you have a cookie INDEX.DAT file loaded and that the Data Location has been set. Press the cookie button on the toolbar and select a cookie URL record. You will see a window at the top of the screen which shows the decoded cookie information.

![Figure 49](image)

- A blue tick next to a cookie record represents a record which has NOT expired when compared to today’s date and time.
- A red circle with a line through it represents a cookie record which HAS expired when compared to today’s date and time.
Cookie Myths & Defences

When cookies first appeared, some Web users were concerned because they thought cookies would track their every move online. Cookies can perform limited tracking when you are browsing Web pages and some persistent cookies can trace your movements from site to site. For example, cookies from Double-Click, a company that feeds targeted Web ads to users, track your surfing to any Double-Click enabled site to make sure that you don't see the same advertisement over and over again.

However, there are a number of myths regarding cookies that need to be dispelled. First of all, cookies are set by the browser and contain text data only. Here are a number of cookie myths that have been raised during a forensic investigation or have been raised as a defence.

- Cookies cannot look into your computer and extract personal information – they cannot find information out about you
- Cookies cannot read, delete or place material on a suspect hard drive
- Cookies cannot hold or execute Viruses or Trojans
- Cookies are NOT written by the web site – they are created by the browser
- Cookies can store personally identifiable information used to identify an individual such as name, e-mail address, home or work address, telephone number and other personal information, but ONLY if YOU provide it to the web site
- A web site cannot gain access to other information on your computer via cookies

Forensic Benefit of Cookies

The forensic examination of cookies may be of benefit to a forensic examination. The cookie may contain usernames, passwords, shopping cart information, preferences, search criteria and other beneficial information. They may indicate the visit to a site and can show access to online storage. They could also show access to password protected sites or areas.

Be aware of 3rd party cookies. The existence of a third party cookie does not necessarily mean the user visited the domain or has any knowledge of the domain. Third party cookies can be deposited through links to external resources.
MSIE Browser Forensic Analysis

Data Extraction

To examine Internet Explorer data, a forensic analysis tool is required. Normally, a forensic image of a suspect hard drive has already been obtained and the next step of the forensic analysis is Internet browser activity. The procedure for Internet Explorer is usually a 5 step process:

- Extraction of Temporary Internet Files folder
- Extraction of Cookies folder
- Extraction of the History folder
- Extraction of History from unallocated space
- Extraction of any other files which may contain Internet History data

We will deal with live internet history for the moment. Extraction from other files and unallocated space will be discussed in another chapter.

Exporting Data

Create an Export folder on your forensic workstation so that all browser data can be extracted. Copy to this folder the entire contents of the History folder, Cookie folder and Temporary Internet Cache Folder. You should end up with the following:

![Image of Export folder contents]

Figure 50
If you examine internet data using the Open All History from folder method, then this must be run against extracted internet data in the same structure as was located on the original disk. You cannot use this method to run against copied out INDEX.DAT files as the forensic software will rename the files. If you wish to examine the history files in that way, then you must select them by using Open History.

Now you are ready to open this data for analysis. Open NetAnalysis and from the File menu select Open All History from Folder.

Navigate to your export folder and select it as follows:

Then select OK. NetAnalysis will then commence a search through all of the folders for INDEX.DAT files. If the procedure is done this way, it makes it easier to identify the original location of the INDEX.DAT file as the path shown is relative to your export folder. This will reflect the original location on the suspect hard drive. It is possible to copy out just the INDEX.DAT files if you are interested in reviewing browser history without rebuilding cached pages.
You are now in a position to update some case information in Case properties (See the Case Properties chapter) and to analyse the user activity.

TIP: It is not advisable to mix history types when conducting analysis. For example, do not mix Internet Explorer History with Netscape History.

When working with multiple profiles and rebuilding web pages, you will need to deal with each user profile separately.
Netscape Browser

Netscape History

In mid-1994, Silicon Graphics founder Jim Clark collaborated with Marc Andreessen to found Mosaic Communications (later renamed to Netscape Communications). Andreessen had just graduated from the University of Illinois, where he had been the leader of a certain software project known as "Mosaic". By this time, the Mosaic browser was starting to make splashes outside of the academic circles where it had begun, and both men saw the great potential for web browsing software. Within a brief half-year period, many of the original folk from the NCSA Mosaic project were working for Netscape, and a browser was released to the public.

Platforms 4.x

Macintosh: 68K, Power Mac  
PC: Win9x, 3.X, NT [Intel and Alpha], 2000/XP  
Unix: AIX, BSDI, HP-UX, IRIX, Linux, OSF, Sparc Solaris, SunOS  
Other: Alpha, OS/2, VAX

Platforms 6.x

Macintosh: OS 8.5-9.x, OSX  
PC: Win95/98/ME, NT/2000/XP  
Unix: AIX, BSDI, HP-UX, IRIX, Linux, OpenVMS, Solaris, Tru64  
Other: BeOS, OS/2

Netscape quickly became a success, and the overwhelming market share it soon had was due to many factors, not the least of which was its break-neck pace of software releases (a new term was soon coined - "internet time" - which described the incredible pace at which browsers and the web were moving). It also created and innovated at an incredible pace. New HTML capabilities in the form of "extensions" to the language were introduced. Since these capabilities were often flashier than what other run-of-the-mill browsers could produce, Netscape's browser helped cement their own dominance. By the summer of 1995, it was a good bet that if you were browsing the Internet, you were doing so with a Netscape browser - by some accounts Netscape had as much as an 80%+ market share.

With the launch of Windows 95 and a web browser of its own (Internet Explorer) in August 1995, Microsoft began an effort to challenge Netscape. For quite a while, Internet Explorer played catch-up to Netscape's continual pushing of the browsing technological envelope, but with one major advantage: unlike Netscape, Internet Explorer was free of charge. Netscape version 2.0 introduced a bevy of must-have breakthrough features (frames, Java, JavaScript and Plug-ins) which helped distance it from the pack, even with its attendant price tag. Mid-1995 to late-1996 was a very busy time for both browsers; it seemed like every week one company or the other was releasing a new beta or final version to the public, each seemingly trying to one-up the other.

But slowly, Internet Explorer gained market share ground. By the fourth generations of both browsers, Internet
Explorer had caught up technologically with Netscape's browser. As time went on, Netscape's market share diminished from its once-towering percentages.

In January 1998, Netscape made an announcement that their browser would thereafter be free and also that the development of the browser would move to an open-source process. This came as wonderful news to many on the Internet. However, the time between this announcement and the actual delivery of Mozilla 1.0 would be long (over 4 years). The process ended up taking much longer than originally anticipated, what with the Netscape/AOL merger and the late-hour decision to integrate an entirely new next-generation HTML rendering engine.

Its initial release of Netscape 6.0 was considered slow and buggy, and adoption was slow to occur. Now that Mozilla has finally reached what it considers to be a significant milestone in its development process (1.0 - which Netscape 7.0 is based on), perhaps those market share usage numbers will increase again.
Netscape Browser Data

Forensic Analysis of Netscape

NetAnalysis currently supports the following versions of Netscape and Netscape Communicator.

- Navigator/Communicator up to 4.8
- Netscape 6, 7 and 8
- Mozilla and Firefox Browsers

Netscape Communicator/Navigator Version 4.8

Netscape Communicator is a proprietary Internet suite produced by Netscape Communications Corporation. Initially released in June 1997, Netscape Communicator 4.0 was the successor to Netscape Navigator 3.x and included more groupware features intended to appeal to enterprises.

In 1998, a major update to the program was released as Netscape 4.5. This included many improvements, mostly to the Messenger email client. By the time version 4.5 was released, Netscape had started the Mozilla open source project and had ceased charging for Communicator.

Technically, "Navigator" was supposed to refer to the browser component alone, and "Communicator" was supposed to refer the suite as a whole, however, due to user confusion, the names were often used interchangeably.

Netscape Communicator was superseded by Netscape 6 in November 2000. However, minor updates to Communicator continued to be issued, culminating in the release of Netscape Communicator 4.8 in August 2002.

Figure 53
Extraction of Netscape 4.8

The following files contain the history, cache, cookie and bookmark data for Netscape Communicator (Navigator) version 4.0 to 4.8. NetAnalysis 1.36 can view

- History Index – netscape.hst
- Cache Index – fat.db
- Cookie Store – cookies.txt
- Bookmark Store – bookmark.htm

A typical location is as follows:

```
..\Netscape\navigator\n..\Netscape\user\{username}\n```

Figure 54

netscape.hst
This is the history file that keeps a record of all Internet Activity.

```
..\Netscape\navigator\cache
..\Netscape\user\{username}\cache
```

fat.db
This is the file which contains the index for the cache. All the files that go to make up a web page are referenced within this file.
**Netscape Version 6 - 8**

On 14th November 2000, AOL released Netscape Version 6 which was based on pre-release Mozilla code.

Given the success of the Mozilla Foundation's standalone Mozilla Firefox browser, the Netscape suite was finally abandoned in favour of a Netscape branded version of Firefox named Netscape Browser, with the first stable version released in 2005. Netscape Browser was not developed in-house by AOL, but instead outsourced to a Canadian software firm, Mercurial Communications. The new browser offered emphasis on anti-phishing features, as well as an ability to switch between the two available rendering engines: Trident (used in Internet Explorer) or Gecko (used in Mozilla).

*Warning: Because the browser can render pages using Trident & Gecko engines, the Internet History and Cache data will end up in two different locations. The Trident rendered data will be recorded and stored in the Internet Explorer history and cache, whilst Gecko rendered pages will be stored in the Mozilla based history and cache. This makes it very difficult for Netscape pages to be rebuilt and history to be tracked. Netscape 8 uses twin engine rendering.*

Users have criticised several aspects of the approach taken, from the fact that using Internet Explorer’s rendering engine subjects it to all the security problems that plague Internet Explorer, to the quality of the user interface design. Some users are also unhappy at the decision to produce only a browser, rather than a full internet suite like previous versions. It remains to be seen whether the browser's distinctions from Firefox will be sufficient to grant it a significant user base.

**Netscape Browser Data Files**

Some locations where the files may be found are as follows, but there may be others:

```text
..\Documents and Setting\username\Mozilla\profiles\user\..\Cache
..\Documents and Setting\username\Mozilla\profiles\user\..
```

*history.dat*

This is the history file which keeps a record of all Internet Activity. The file structure of this file is completely different from Netscape Communicator.

The cache is located in four files as follows:

```text
  _CACHE_001_
  _CACHE_002_
  _CACHE_003_
  _CACHE_004_
```

At the moment, NetAnalysis does not support Netscape 6-8 cache extraction, this will be added shortly.
Mozilla Browser Data

Mozilla Profile Location

In Windows NT based Operating Systems, the Mozilla profile can be located as follows:

\%APPDATA\%Mozilla\Profiles\[profile name]\[random string\].slt\n
The \%APPDATA\% folder is usually located as follows:

C:\Documents and Settings\[Profile Name\]Application Data\n
Mac OS

With Macintosh OS9, the profile data is located:

disk::Documents::Mozilla::Profiles::[profile name]::[random string].slt/

With Macintosh OSX, the profile data is located:

~/Library/Mozilla/Profiles/[profile name]/[random string].slt/

Linux

With Linux, the profile data is located:

~/.mozilla/[Linux Login Name]/[random string].slt/

Random SLT Folder

As an additional security measure Mozilla adds a random string to the path of the profile. This makes it more difficult for the authors of malicious code to identify the location of the profile.

Files in the Mozilla profile directory

The following information outlines the various files and folders located in the Mozilla profile. The folders are as follows:

<table>
<thead>
<tr>
<th>Folder Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache</td>
<td>Cache files (temporary internet files) [about:cache]</td>
</tr>
<tr>
<td>Cache.Trash</td>
<td>Temporary cache directory. Only used when deleting the real Cache directory.</td>
</tr>
<tr>
<td>calendar</td>
<td>Calendar files</td>
</tr>
<tr>
<td>cert8.dir</td>
<td>Client Certificate database. [Edit -&gt; Preferences -&gt; Privacy &amp; Security -&gt; Certificates -&gt; Manage Certificates]</td>
</tr>
</tbody>
</table>
### Chrome
- User defined styles and installed themes and extensions

### Extensions
- Extensions directory

### ImapMail
- IMAP mail

### Mail
- Local Folders and POP3 mail

### News
- Newsgroups

### Security
- Security files (Mac OS only)
- Security Modules - "secmod.db"
- Certificates - "cert7.db" and "cert8.db"
- Key Database - "key3.db"

### Nim
- Used in AIM. (Netscape 6/7 only)

---

The following information outlines the various files located in the Mozilla profile.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[number].s</td>
<td>Password data [Tools -&gt; Password Manager -&gt; Manage Stored Passwords]</td>
</tr>
<tr>
<td>[number].w</td>
<td>Fill Form data [Tools -&gt; Form Manager -&gt; Edit Form Info]</td>
</tr>
<tr>
<td>ac-weights.txt</td>
<td>Autocomplete training weight file</td>
</tr>
<tr>
<td><em>appreg</em> and <em>registry</em></td>
<td>Application Registry, Mozilla XPCOM registry. Files where Mozilla keeps track of its installed pieces (Linux only)</td>
</tr>
<tr>
<td>abook.mab</td>
<td>Personal address book [Window -&gt; Address book -&gt; Personal Address Book]</td>
</tr>
<tr>
<td>bookmarks.html</td>
<td>Bookmarks [Bookmarks -&gt; Manage Bookmarks]</td>
</tr>
<tr>
<td><em>cert7.db</em> and <em>cert8.db</em></td>
<td>Client Certificate database [Edit -&gt; Preferences -&gt; Privacy &amp; Security -&gt; Certificates -&gt; Manage Certificates] - (cert7.db is the old format, cert8.db is the new)</td>
</tr>
<tr>
<td><em>Component Registry</em> or <em>component.reg</em></td>
<td>Mozilla XPCOM registry. Files where Mozilla keeps track of its components.</td>
</tr>
<tr>
<td>cookies.txt</td>
<td>Cookies [Tools -&gt; Cookie Manager -&gt; Manage Stored Cookies]</td>
</tr>
<tr>
<td>cookperm.txt</td>
<td>Obsolete Replaced by hostperm.1. Cookie and Image permissions [Tools -&gt; Cookie Manager -&gt; Manage Stored Cookies -&gt; Cookie Sites] and [Tools -&gt; Image Manager -&gt; Manage Images Permissions]</td>
</tr>
<tr>
<td>compatibility.ini</td>
<td>Compatibility listing. Automatically generated file</td>
</tr>
<tr>
<td>components.ini</td>
<td>Components listing. Automatically generated file</td>
</tr>
<tr>
<td>compreg.dat</td>
<td>Component registration listing. automatically generated file</td>
</tr>
<tr>
<td>downloads.rdf</td>
<td>Download Manager data [Tools -&gt; Download Manager]</td>
</tr>
<tr>
<td>formhistory.dat</td>
<td>Autocomplete data for Forms (Firefox Only)</td>
</tr>
<tr>
<td>history.dat</td>
<td>URL history [Go -&gt; History]</td>
</tr>
<tr>
<td>history.mab</td>
<td>Collected Address book [Window -&gt; Address Book -&gt; Collected Addresses]</td>
</tr>
<tr>
<td>hostperm.1</td>
<td>New Cookie and Image permissions [Tools -&gt; Cookie Manager -&gt; Manage Stored Cookies -&gt; Cookie Sites] and [Tools -&gt; Image Manager -&gt; Manage Images Permissions]</td>
</tr>
<tr>
<td>File Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>key3.db</td>
<td>Key database [Edit -&gt; Preferences -&gt; Privacy &amp; Security -&gt; Certificates -&gt; Manage Certificates]</td>
</tr>
<tr>
<td>&quot;listing-downloaded.xml&quot; and &quot;listing-uploaded.xml&quot;</td>
<td>Roaming profiles files [Edit -&gt; Preferences -&gt; Roaming User]</td>
</tr>
<tr>
<td>localstore.rdf</td>
<td>Defines default windows settings. Saved persistent values. Info about toolbars, size, positions</td>
</tr>
<tr>
<td>mailViews.dat</td>
<td>Defines your current message views [View -&gt; Customize...]</td>
</tr>
<tr>
<td>mimeTypes.rdf</td>
<td>Defines recognized file mime types for the Helper App [Edit -&gt; Preferences -&gt; Navigator -&gt; Helper Applications]</td>
</tr>
<tr>
<td>panacea.dat</td>
<td>Mail folder cache. Summary of certain fields in all the mail/news databases, mostly used to display the folder pane</td>
</tr>
<tr>
<td>panels.rdf</td>
<td>Information about which panels that will be displayed in the sidebar</td>
</tr>
<tr>
<td>&quot;parent.lock&quot; or &quot;lock&quot; or &quot;.parentlock&quot;</td>
<td>Lock file that indicates that the profile is in use</td>
</tr>
<tr>
<td>persdict.dat</td>
<td>Personal dictionary entries. Use by the spellchecker.</td>
</tr>
<tr>
<td>prefs.js</td>
<td>All your settings [about:config]</td>
</tr>
<tr>
<td>prefs.bak</td>
<td>Backup of your prefs.js file</td>
</tr>
<tr>
<td>search.rdf</td>
<td>Information about your search plug-ins</td>
</tr>
<tr>
<td>signons.txt</td>
<td>Remembered usernames and passwords</td>
</tr>
<tr>
<td>storage.sdb</td>
<td>sqlite database used as mozStorage profile db</td>
</tr>
<tr>
<td>training.dat</td>
<td>Training data for the Bayesian filter</td>
</tr>
<tr>
<td>URL.tbl</td>
<td>Excluded Fill Form data sites [Tools -&gt; Form Manager -&gt; Manage Sites -&gt; Forms Never Saved]</td>
</tr>
<tr>
<td>url-data.txt</td>
<td>URL history data capture file</td>
</tr>
<tr>
<td>user.js</td>
<td>User setting which will be written into prefs.js after Mozilla is started</td>
</tr>
<tr>
<td>xpti.dat</td>
<td>XPCOM typelib information catalog</td>
</tr>
<tr>
<td>&quot;XUL.mfl&quot; or &quot;XUL FastLoad File&quot; or &quot;XUL.mfasl&quot;</td>
<td>XUL fast load file. Contains precompiled chrome and JavaScript.</td>
</tr>
<tr>
<td>*.ics</td>
<td>Old Calendar Files. New files are in the &quot;calendar&quot; directory.</td>
</tr>
</tbody>
</table>
Case Properties

Updating New Case Properties

The Case Properties window can be accessed from the toolbar or File >> Case Properties. This will allow you to store case specific data which will be stored in the NetAnalysis workspace file. The first tab shows the Case information as follows:

To be able to examine and rebuild extracted cache data along with cookie data, NetAnalysis needs to know the location of the exported data. If you have selected this data by using the open All History from folder method, the paths should have been automatically identified for you. If this is not the case, they can be manually set from the following location:

Remember that if you change the location of this export folder, you will need to update the path data to reflect the new location.
The final tab allows you to enter some information about this case. You also have a choice as to whether the information is printed at the end of the advanced report. Mainly, this area is used as a reminder for the investigator.

![Case Properties](image)

Figure 57
System Log

Viewing the System Log

During the process of extracting internet history, NetAnalysis keeps a system log which holds information such as who the software is licensed to, the date/time the analysis commenced, the Investigator, the time zone settings and the version of NetAnalysis used. The system log can be saved as a separate text file if required.

The system log also records the path and name of every file extracted. As a further check, NetAnalysis will also record the apparent time zone settings when it examines a Daily INDEX.DAT file. It is not unusual for a user to have the time zone set incorrectly, or to change the time zone settings at various times during the use of the system. Expect to see time zone settings change from Standard Time to Daylight Saving Time over the course of a year.

Figure 58
URL Record Types (SCHEME)

Understanding the Different Record Entries

Now that you have some data to examine, you need to be able to understand each type of record entry or scheme. The different entries are as follows:

**URL**
This entry is a standard History URL (Uniform Resource Locator) and is explained in depth in the HTTP chapter on Page 81.

**Cached**
This record identifies entries from the Cache or Temporary Internet Files. This can give an indication as to which sites have been visited by the user, but may not necessarily be the case. A page can be made up of pictures and resources located on other web servers. All cached entries are identified in this way regardless of scheme.

**Secure**
This record identifies History entries which have shown access to SSL / HTTPS pages.

**File (Local File Access)**
This record shows access to a local file on the local system and is not a file viewed over the Internet. The access may be opening or saving a file. Internet Explorer also keeps track of files that have been opened or saved by the user on the local system. Normally, this entry will reflect files which have been opened that have a corresponding shell entry in the registry. For example, the extension DOC may be associated with Microsoft Word. Any Word documents opened by the local user will be recorded in the history files. There is a common
myth that the file would need to be opened within Internet Explorer. This is not the case. Any registered file
type opened with the associated application will result in an entry, much the same way as link files are created.

**FTP**
This record will show access to an FTP site via FTP protocol through Internet Explorer.

**E-Mail**
This record will show mailto: entries or access to Microsoft Outlook on the local system. Outlook is closely
linked with Internet Explorer as it has a built in web browser and shares common components.

**News**
This will show NNTP (Network News Transport Protocol) entries and access to newsgroups via a browser.

**Cookie**
This record will show cookie entries and will only be extracted from an INDEX.DAT file that records cookie
data.

**Host**
Found only within Daily and Weekly INDEX.DAT files, this entry is used to build the hierarchical structure of
History pages when the history is viewed from within Internet Explorer. Viewing the HOST entries will allow
you to see the web sites that Internet Explorer itself would show as a history entry.

**Help**
Many modern help files are now compiled HTML documents. As such, any access to compiled HTML (CHM)
or HTML help documents get recorded as entries within the INDEX.DAT file.

**Java**
Many web pages use java scripting. These entries show access to particular JavaScript applications stored
within JavaScript (*.JS) pages.

**RES**
This entry shows access to a resource embedded within another file. Many of the error pages you see shown
by Internet Explorer are actually stored as a resource within a DLL (Dynamic Link Library). Internet Explorer
stores its own 404 page within a DLL.

**Redirect**
These entries are server side redirects. This is NOT evidence of a pop-up window, but a page which has
asked the browser to re-locate. This is explained in more depth on Page 98.

**Leak**
This entry is similar to a URL entry and appears in the cache INDEX.DAT. There is much debate about the
actual purpose of this entry. From a forensic point of view, the entry is similar to a normal URL entry and
reflects the access to a site or resource.

**Unknown**
This entry appears when NetAnalysis cannot identify the type of record or scheme.
NetAnalysis Column Headers

Understanding Column Headers

When data is loaded into the NetAnalysis software, it is presented to the Forensic Computer Analyst in a data grid. There are 20 different fields available. The column headers are explained as follows:

Type
This is discussed in the previous chapter. This field identifies the type of internet record whether that is a cache entry, a history entry, a file from the local machine or cookie entry. See the chapter on understanding URL Record Types. Another way to describe this is the URL Scheme.

Tag
This column will show which records have been tagged by the forensic computer analyst.

Last Visited
This is the Date / Time that the user last visited a page. The default is UTC, but can be altered prior to analysis for the properties window. From version 1.35 of NetAnalysis, this column should always be the standard time of the time zone set, so if the time zone is set to UTC, the Last visited Date / Times should also be UTC. Please read in conjunction with the chapter on Understanding Date / Time issues and Time Zones.

Secondary Date
This is also an important date and will show different things depending on which INDEX.DAT or history file type is being examined. Please read in conjunction with the chapter on Understanding Date / Time issues.

User
This column will show the user name or profile name that was logged on when the web site was accessed. When recovering data from unallocated space, it is often possible to recover data from deleted profiles.

With NT based operating systems, the name appearing here relates to the profile name of the current logged on user. With Windows 95/98 and ME, the name relates to the username entered into the login box. As we know, it is possible to place any name or data in this box and it logs that user on. If the user at this point decides to use a password, the operating system will prompt to verify the password. However, no password is required.

Status
This column has a different meaning depending on the type of Internet History being examined. With Internet Explorer Daily INDEX.DAT files, this column will show the offset from UTC for a specific entry. In Mozilla and Netscape 6/7/8 history records, this column will show if the URL record has been recorded as a Typed URL. This type of record usually reflects data that was typed into the address bar. With Internet Explorer, this data is stored within the registry and not the history file.
Hits
This column has a different meaning depending on which type of history you are examining and which type of history file the entry is related. This column can reflect the number of visits to a particular site or page or with cached entries it will show the number of times an image or resource has been retrieved from cache.

- Master History - Examination of the Master INDEX.DAT file in MSIE will show the HIT count for web pages visited. It would be hard for a user to explain that a web site containing indecent images of children appeared in the history as a result of a pop-up window, if the INDEX.DAT hit rate showed that it had been visited on numerous occasions.
- Daily History – Examination of the Daily INDEX.DAT file in MSIE will show the number of visits to that page or resource on a particular day, at that time.
- Weekly History – Examination of the Weekly INDEX.DAT in MSIE will show the number of visits to that page or resource during the weekly browsing period.
- Cache – Examination of the INDEX.DAT file in MSIE will show the number of times that file/page or resource has been retrieved from the cache.

With binary index files where a hit count is not recorded – the hit value will be recorded as 1. Hit counts within Internet History files should only really be used as a guide. The most accurate count is within the MASTER INDEX.DAT file for Internet Explorer.

URL
This is the Internet URL – further information regarding the make-up of URL records can be found in the HTTP Chapter on Page 81.

Host
The host column shows the domain in which the web page or resource actually resides. This will reflect the top level domain.

Web Page Title
This column will show the name of the page as it would have appeared in the title bar of Internet Explorer.

Search Engine Criteria
NetAnalysis will try and extract the search criteria used if it identifies a URL from a search engine.

Cache Folder
This column will show the name of the cache folder in which the page/picture or resource was saved. For Internet Explorer URLs from unallocated space, it is not possible to identify the cache folder, as the data resides at the start of the complete INDEX.DAT file. In this case, the cache folder will remain empty.

URL Filename
This column will show the cache file name as had been changed by the browser. Internet Explorer always adds a square bracket to the end of the name with an identifying counter. This counter is incremented if a different file with the same name is added to the same cache folder. I.e. picture.jpg becomes picture[1].jpg.
Internet Explorer is not the only software package to save data to the cache. Outlook Express uses the cache as a temporary store for e-mail messages and binary attachments. In this case, naming convention is Wbk***.tmp (without square bracket), where the asterisk is replaced by a random hex digit. i.e. wbk12E.tmp there will be no INDEX.DAT entries for these files.

Exists
This column will identify live cache entries when the index entry is compared to files stored in the cache. Verification of the cache can be completed as follows: Tools >> Verify Live Cache Entries.

HTTP Response Header
This column shows the HTTP response header as returned by the web server. This information is stored in the cache INDEX.DAT file. HTTP Header information is discussed at length on page 81.

Source History File
This column will show the full path to the internet history record from which the data has been extracted.

File Offset
This column will show the decimal offset within the source history file where the URL record was found. The Hexadecimal offset will be shown in the Status Bar.

Index type
This column will show from which INDEX.DAT file type the record was extracted. This could be History, Daily History, Weekly History, Cache or Cookie.

History Version
This column will identify the version of Internet History from which the record was extracted. History extracted from unallocated space will also be identified as such.

Comment
This column will show the bookmark comment as set by the forensic examiner.

URN
This column shows a unique record number, which always remains static. This record number can be used to identify a specific record.
Filtering/Searching and Sorting

Filtering Internet Records

It is important to be able to quickly identify the internet history records that can prove your case. NetAnalysis has a number of different ways to do this. One of the easiest ways is to apply a filter that only returns records that contain the keyword.

Click on the filter button, press F8 or select Search >> Select Filter Criteria from the menu. FILTERS can be removed by pressing the REFRESH button or F5.

This will show the Filter Window as below:

![Filter Window](image)

From here you can select the field to filter and type a key word into the Filter Text box. When you press OK, NetAnalysis will filter the required field and return all the URL records which contain your keyword.

To remove the filter press F5. The initial default sort field is the Last Visited date/time, which is assorted in Ascending Date order.

If you do not wish to filter the records and just want to quickly find the first URL record containing a key word, just select the find button or press F7. This will allow you to find the first Internet History record which contains that word. Subsequent hits can be identified by pressing F3. (F2 will select previous entries).

![Find First Window](image)
Advanced Filtering

The Filter menu has a number of options to quickly identify certain record types. Figure 62 below shows the Filter menu and the Advanced Filter menu:

The Advanced filter menu has filters for the following:

- Internet Records containing references to Indecent Images of Child Abuse
- Internet Records containing Possible Search Engine Activity
- Internet Records containing Searches using GOOGLE
- Internet Records containing Google Desktop Searches
- Internet Records containing plain text Usernames & Passwords
- Internet Records containing User Login Information
- Internet Records showing access to possible Online Storage Sites
- Records showing access to files on a floppy diskette
- Internet Records showing files previewed within the Kazaa File Sharing software
- Internet Records containing JPEG, PNG, GIF, BMP or ART graphic files
- Internet Records containing ZIP, RAR and other types of Archive Files
- Internet Records containing MPEG, AVI, MOV, RAM and other types of Video Files
- Internet Files which have been returned as GZIP compressed pages

It is also possible to filter records based on index type. This is accessed from the Filter menu. The possible filter options are shown in Figure 63.
Multiple Keyword Searching

NetAnalysis has the ability to perform keyword searches on your own keyword lists. These lists can be imported and exported. The keyword lists are simple text files with a keyword on each line. The lists can also be commented by using two forward slash characters //.

NetAnalysis comes with a few examples. The lists are stored in the “Keyword Lists” folder in your NetAnalysis folder. To open the keyword list, either hit F4 or select User Keyword List from the Searching Menu.

You can add up to 90 keywords. The keyword list can be run against the Internet history column, the Page Title column and the URL File Name column.

When you add your keywords, you can specify how you want them handled. You can set them to identify URLs where ALL your keywords must exist or where ALL of your keywords can exist (AND/OR searching).

If you wanted to identify Hotmail account activity where a message was viewed – you could add the following keywords:

```
.hotmail.
?curmbox
getmsg?
```

You would set these to “Must contain all” as you are interested in any URL records that will have all three keywords present.
This is a powerful feature as you could maintain your own lists that identified areas such as

- Indecent Images of Children
- Adult Pornography
- Inappropriate web sites
- Web based e-mail activity
- Login, username and passwords
- Search engine criteria
- Online storage sites
- Activity using evidence eliminator, cryptography or steganography
- Web based text messaging

**Column Filter Bar**

Another powerful feature to enable record filtering is the Column Filter Bar. The column filter bar can be accessed from the toolbar or by selecting Column Filter Bar from the View menu. Criteria can be entered in multiple column headers and then filtered out. This is a quick and easy way to filter on multiple fields at once. Hitting enter will activate the filter. Pressing F5 will remove the filter.

Figure 65

![Column Filter Bar](image)
SQL Query Filtering

Filtering Records with SQL

NetAnalysis has a powerful facility to enable the forensic computer analyst to filter data using custom SQL Queries. Queries can be designed to filter out specific data and then saved for later re-use. The SQL Query window can be accessed by pressing CTRL + F4 or selecting Build SQL Query from the Searching Menu. The window will appear as follows:

![SQL Query Window]

The example query above will select all Internet Records showing MAIL activity that occurred on a Wednesday. The resulting data will be ordered by the Last visited column. NetAnalysis comes with a number of example queries that demonstrate how this feature works.

There are many resources available to teach SQL. It is currently beyond the scope of this manual to teach SQL programming and syntax.

Another example would be as follows:

```sql
SELECT * FROM History
WHERE History.InternetHistory LIKE 'ftp://*'*@*'
```

The above query will filter out any records where a username and password have been used to access a web or FTP site. The query uses the asterisk wild card character.

For example:

```
ftp://administrator:letmein@ftp.download-my-files.com
```
Processing Evidence

Tagging Records

As you examine hundreds of URLs from a suspect case, you may want to tag individual URL records for further examination, or for producing an evidential report. Tagging a URL is as simple as hitting the space bar, selecting F11 or selecting Tools >> Tag URL record. In normal view, the highlighted row will be set to bold to indicate a tagged record. A tick will also be placed in the TAG column, along with an indicator in the Status Bar. Figure 67 below shows a number of tagged records.

Once you have your records tagged, they can be filtered by selecting Filter >> Filter Tagged URLs or by pressing F9. If you print or preview a report at this stage, you will obtain a snapshot of the currently filtered data. To clear the filter, press F5.

To remove or add tags from/to the currently filtered data, select:

- Tools >> Remove Tags from Current Filtered URL Records (This will only remove tags from the records currently filtered on screen)  If you wish to remove all tags, make sure you are viewing the entire data set by clearing filters with F5.
- Tools >> Tag Current Filtered URL Records (This will add tags to all of the currently filtered records)

Bookmarks & Comments

As you examine your data, you may find URL records that are of evidential value or records that need to be explained or commented. To allow you to do this, NetAnalysis has a feature for bookmarking and commenting individual URL records.
Pressing the ENTER key or select TOOLS >> Add / Edit Comment, will show the URL Record Comment window:

![URL Record Comment](image)

Figure 68

In the bottom pane, type the information you need to either explain or highlight this URL record. The information from the bookmark will be visible in Comment column and in the Advanced Report. Once the comment has been added, the individual record will be highlighted with a bookmark icon as shown below.

![Google search for plastic explosive](image)

Figure 69

The advanced report will also show your comment.

![Forensic Internet History Report](image)

Figure 70

To filter commented records:

- Filter >> Filter Records with Comments or press CTRL + F9
- Filter >> Filter Tagged and Comments (Will filter tagged records where comments have been added)
Exporting Data

Export Format

NetAnalysis supports exporting the evidence in a number of different formats.

- Tab delimited text file (will only export current filtered records)
- Comma delimited text file (will only export current filtered records)
- Microsoft Access database (entire recordset)
- HTML (will only export current filtered records)
- Adobe PDF

With the exception of Adobe PDF format and Microsoft Access database, only the selected columns will be exported. If you want to export only specific column data, hide the other columns by selecting them from the Column menu.

Adobe PDF format is available from any of the print functions such as the Report menu. This is discussed in more detail in the chapter covering evidence processing.

The export functions can be found on the File menu under File >> Export History As.
Deleted History Extractor

Extracting Deleted Internet History

Another vital source of evidence is from Internet History that has been extracted from unallocated clusters. Because of the way Internet Explorer manages history and cache INDEX.DAT files, you should be able to find an ample amount of deleted history. Even if the user is using evidence elimination software on a regular basis, it is difficult to eradicate fragments of Internet History.

NetAnalysis comes with an extractor (HSTEX) which is designed for carving out individual Internet Explorer (and Safari) URL records from a number of different sources. They may be:

- Linux DD image formats
- Physical device
- Logical device
- Exported binary file Unallocated Space Files
- Write Protected Physical Devices
- Write Protected Logical Devices
- Mounted Paraben P2 Explorer Disks (See section below)
- Mounted Encase VFS Files/Disks
- Mounted Encase PDE Devices
- Mount Image Pro - Mounted Physical/Logical Disks
- Paraben Forensic Sorter Files
- Paraben Forensic Replicator Images
- Vogon Images
- Generic Flat File Images
- iLook Extracted Unallocated Space
- Mounted compressed image format

This is actually quite a large choice of source file formats, so the forensic investigator will need think carefully about what type of evidence he/she wishes to identify.

**TIP:** Remember that no matter which file/device you select, HSTEX will search for individual URL records from the first sector to the last. There is NO file size limit on the files it can handle. Obviously, if you point HSTEX at a device other than mounted/exported unallocated clusters, you will get LIVE and DELETED internet history. This may not be the best option.

*HSTEX will shortly be updated so that it can extract data directly from unallocated clusters.*
Unallocated Clusters

The recommended approach for full analysis of live and deleted Internet History is to extract the LIVE data in the same file/folder structure as the suspect system. To complete the examination, you will need all of the URL records available from unallocated clusters.

The easiest method is to mount your forensic image using mounting software such as Guidance Software VFS. HSTEX can then be pointed directly at the “virtual file” containing the unallocated clusters.

Please note that when viewing data extracted from unallocated clusters, NetAnalysis currently shows the offset of the data within the extracted file and not the unallocated clusters themselves. This is also true for extracting data across physical/logical devices and image file formats. The next version of HSTEX will be capable of showing the offset of the data on the original device.

An alternative method is to copy out the unallocated clusters for processing separately. If this method is used, please ensure that the data is extracted as ONE full complete file. DO NOT split the file into smaller chunks. HSTEX can handle a file of any size. By splitting into smaller chunks, you run the risk of missing data at file boundaries.

To extract data from a binary file or mounted virtual file, click the button highlighted in Figure 72 below and select the file.

Figure 72

The output folder will be where the extracted data will be saved. HSTEX will extract the data and save it in multiple files (up to 25Mb in size for convenience).

Don’t forget to select what type of history you wish to extract. Other formats will be available soon.
Physical / Logical Devices

It is also possible to extract history from physical and logical devices. If you decide to extract history from a write protected or mounted physical or logical device, please remember that HSTEX will search the device in a sequential manner from sector 0. It will not walk through the cluster chains. This means that data in non-contiguous clusters may not be extracted.

HSTEX will be updated to add support for extracting data from unallocated clusters directly, without having to use VFS or extract the data.

To extract from a physical/logical device, click the button highlighted in Figure 73, below.

Figure 73

This will present a window showing you all the available physical and logical devices. Select one of the devices as shown in Figure 74.

Figure 74
Hypertext Mark-up Language

Introduction to HTML and Web Pages

HTML is the coding language used to create Hypertext documents for use on the World Wide Web. HTML looks a lot like old-fashioned typesetting code, where you surround a block of text with codes that indicate how it should appear. The "hyper" in Hypertext comes from the fact that in HTML you can specify that a block of text, or an image, is linked to another file on the Internet.

This is a short introduction to HTML. Having an understanding of how web pages are created and how HTML works, will assist you in the forensic analysis of browser data. Many people still write HTML by hand using tools such as Notepad, or a simple text editor.

Every HTML document needs a title. To create a simple HTML web page, we need nothing more than notepad. HTML is just a series of tags that are integrated into a text document. They are a lot like stage directions - silently telling the browser what to do, and what props to use.

HTML tags are usually English words or abbreviations (such as "head" for the header or "p" for paragraph), but they are distinguished from the regular text because they are placed in small angle brackets. Remember that every tag is opened and closed. An example of the header section is as follows:

```html
<head>
  ...
</head>
```

As you can see the head section is opened using angled brackets and closed with a forward slash. The basic HTML page begins with the tag `<html>` and ends with `</html>`. In between, the file has two sections - the header and the body.

The header - enclosed by the `<head>` and `</head>` tags - contains information about a page that won't appear on the page itself, such as the title. The body - enclosed by `<body>` and `</body>` - is where the main page data resides. Every thing that appears on the page is contained within these tags.
A Sample Web Page

To create a simple web page, open notepad and type the following.

```html
<html>
<head>
<title>This is where the Page Title would go</title>
</head>
<body>
<h1>This is the Main Header</h1>
<p>This is an example of a simple page with one line and a title.</p>
</body>
</html>
```

Save the page as test.html and then double click it to launch it in a browser window. When your default browser is launched, the page should display as follows:

![Figure 75](image)

**This is the Main Header**

This is an example of a simple page with one line and a title.

**Headings**

If you have used Microsoft Word, you will be familiar with the built in styles for headings of differing importance. In HTML there are six levels of headings. H1 is the most important or highest level going all the way down to H6 as the least important.

**Images**

Images can be used to make Web pages distinctive and greatly help to get a message across. Images are extremely important to forensic examiners, in particular when investigating offences involving indecent images of children. The simple way to add an image is to use `<img>` tag. For example, let’s assume we have a JPEG picture called CAR.JPG and it is stored in the images folder. The image is 149 pixels wide by 112 pixels high.

To enable the page to display the picture, we have to add the following code to the page:

```html
<p><img border="0" src="images/car.jpg" width="149" height="112"></p>
```
This line has an additional field entitled border. Setting the border value to anything other than zero will place a black line around the outside of the image. The higher the value, the thicker the border will be. The width and height aren’t strictly necessary but help to speed the display of your Web page. You can add a short description with the alt field.

```html
<p><img border="0" src="images/car.jpg" width="149" height="112" alt="This is a BMW M3"></p>
```

The web page will look now look like this:

```html
<html>
<head>
<title>This is where the Page Title would go</title>
</head>
<body>
<h1>This is the Main Header</h1>
<h2>This is a second level heading</h2>
<h3>This is a third level heading</h3>
<p>This is an example of a <em><b>simple</b></em> page with one line and a title.</p>
<p><img border="0" src="images/car.jpg" width="149" height="112" alt="This is a BMW M3"></p>
</body>
</html>
```

From a Web Browser – the page will look like this:

**This is the Main Header**

**This is a second level heading**

**This is a third level heading**

This is an example of a *simple* page with one line and a title.

![BMW M3](images/car.jpg)

Figure 76
Hyperlinks

What makes the Web so effective is the ability to define links from one page to another, and to follow links at the click of a button. A single click can take you right across the world! Hyperlinks are defined with the `<a>` tag. For example, if we wanted to add a simple hyperlink to our page which would take the user to Google, we could add the following:

```html
<p><a href="http://www.google.com">This is a hyperlink to Google</a></p>
```

We could also create a hyperlink from our image as follows:

```html
<a href="http://www.bmw.co.uk">
<img border="0" src="images/car.jpg" width="149" height="112" alt="This is a BMW M3"></a></p>
```

The final web page looks like this:

This is the Main Header

This is a second level heading

This is a third level heading

This is an example of a simple page with one line and a title.

This is a hyperlink to Google

Figure 77
The Hypertext Transport Protocol

Understanding HTTP Protocol

The Hypertext Transport Protocol (HTTP) is the common language that web browsers and web servers use to communicate with each other on the Internet.

URLS

During our examination of HTTP, we will often be referring to URLs, or Uniform Resource Locators. In web terms, a resource represents anything available on the web, whether it is an HTML page, an image, a CGI script, etc. URLs provide a standard way to locate these resources on the Web. URLs are not specific to HTTP; they can refer to resources in many protocols. The following information refers to HTTP URLs.

HTTP URLs consist of a scheme, a host name, a port number, a path, a query string, and a fragment identifier, any of which may be omitted under certain circumstances.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Host</th>
<th>Port</th>
<th>Path</th>
<th>Query</th>
<th>Fragment</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.digital-detective.co.uk:80/cgi/calender.cgi?month=July#week3">http://www.digital-detective.co.uk:80/cgi/calender.cgi?month=July#week3</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 78

HTTP URLs contain the following elements:

Scheme

The scheme represents the protocol, and for our purposes will either be http or https. Https represents a connection to a secure web server (Secure Socket Layer).

Host

The host identifies the machine running a web server. It can be a domain name or an IP address, although it is a bad idea to use IP addresses in URLs and is strongly discouraged. The problem is that IP addresses often change for any number of reasons: a web site may move from one machine to another, or it may relocate to another network. Domain names can remain constant in these cases, allowing these changes to remain hidden from the user.

Port number

The port number is optional and may appear in URLs only if the host is also included. The host and port are separated by a colon. If the port is not specified, port 80 is used for http URLs and port 443 is used for https URLs.

It is possible to configure a web server to answer other ports. This is often done if two different web servers need to operate on the same machine, or if a web server is operated by someone who does not have sufficient rights on the machine to start a server on these ports (e.g., only root may bind to ports below 1024
on Unix machines). However, servers using ports other than the standard 80 and 443 may be inaccessible to users behind firewalls. Some firewalls are configured to restrict access to all but a narrow set of ports representing the defaults for certain allowed protocols.

**Path information**
Path information represents the location of the resource being requested, such as an HTML file or a CGI script. Depending on how your web server is configured, it may or may not map to some actual file path on the system.

Note that the URL for a script may include path information beyond the location of the script itself. For example, say you have a CGI at:

```
http://localhost/cgi/browse_docs.cgi
```

You can pass extra path information to the script by appending it to the end, for example:

```
http://localhost/cgi/browse_docs.cgi/docs/product/description.txt
```

Here the path `docs/product/description.txt` is passed to the script.

**Query string**
A query string passes additional parameters to scripts. It is sometimes referred to as a search string or an index. It may contain name and value pairs, in which each pair is separated from the next pair by an ampersand (`&`), and the name and value are separated from each other by an equals sign (`=`).

Query strings can also include data that is not formatted as name-value pairs. If a query string does not contain an equal's sign, it is often referred to as an index. Each argument should be separated from the next by an encoded space (encoded either as `+` or `%20).

**Fragment identifier**
Fragment identifiers refer to a specific section in a resource. Fragment identifiers are not sent to web servers. Instead, the browser fetches a resource and then applies the fragment identifier to locate the appropriate section in the resource. For HTML documents, fragment identifiers refer to anchor tags within the document:

```
<a name="anchor">Here is the content you require...</a>
```

The following URL would request the full document and then scroll to the section marked by the anchor tag:

```
http://localhost/document.html#anchor
```

Web browsers generally jump to the bottom of the document if no anchor for the fragment identifier is found.
Absolute and Relative URLs

Many of the elements within a URL are optional. You may omit the scheme, host, and port number in a URL if the URL is used in a context where these elements can be assumed. For example, if you include a URL in a link on an HTML page and leave out these elements, the browser will assume the link applies to a resource on the same machine as the link. There are two classes of URLs:

Absolute URL
URLs that include the hostname are called absolute URLs. An example of an absolute URL is:

```
http://localhost/cgi/script.cgi
```

Relative URL
URLs without a scheme, host, or port are called relative URLs. These can be further broken down into full and relative paths:

Full paths
Relative URLs with an absolute path are sometimes referred to as full paths (even though they can also include a query string and fragment identifier). Full paths can be distinguished from URLs with relative paths because they always start with a forward slash. Note that in all these cases, the paths are virtual paths, and do not necessarily correspond to a path on the web server's file system. An example of an absolute path is:

```
/index.html
/graphics/main.jpg
/test/graphics/example.png
```

Relative paths
Relative URLs that begin with a character other than a forward slash are relative paths. Examples of relative paths include:

```
script.cgi
/../images/picture.jpg
../../graphics/image.gif
```

URL Encoding

Many characters must be encoded within a URL for a variety of reasons. For example, certain characters such as ?, #, and / have special meaning within URLs and will be misinterpreted unless encoded. It is possible to name a file doc#2.html on some systems, but the URL http://localhost/doc#2.html would not point to this document. It points to the fragment 2.html in a (possibly nonexistent) file named doc. We
must encode the # character so the web browser and server recognise that it is part of the resource name instead.

Characters are encoded by representing them with a percent sign (%) followed by the two-digit hexadecimal value for that character based upon the ISO Latin 1 character set or ASCII character set (these character sets are the same for the first eight bits). For example, the # symbol has a hexadecimal value of 0x23, so it is encoded as %23.

The following characters must be encoded:

- Control characters: ASCII 0x00 through 0x1F plus 0x7F
- Eight-bit characters: ASCII 0x80 through 0xFF
- Characters given special importance within URLs: ; / ? : @ & = + $ ,
- Characters often used to delimit (quote) URLs: < > # % "
- Characters considered unsafe because they may have special meaning for other protocols used to transmit URLs (e.g., SMTP): { ] ` ^ "

Additionally, spaces should be encoded as + although %20 is also used. As you can see, most characters must be encoded; the list of allowed characters is actually much shorter:

- Letters: a-z and A-Z
- Digits: 0-9
- The following characters: - _ . ! ~ * ' ( )

It is actually permissible and not uncommon for any of the allowed characters to also be encoded by some software. Thus, any application that decodes a URL must decode every occurrence of a percentage sign followed by any two hexadecimal digits.

Some URL Encoding examples are shown below:

1. http://www.google.com
2. http://%77%77%77%2E%67%6F%6F%67%6C%65%2E%63%6F%6D
4. http://%36%36%2E%31%30%32%2E%39%39%39
5. http://1113983331
6. http://%31%31%31%33%39%38%33%33%31

Example 1 (http://www.google.com)
This shows the standard representation of a URL.

Example 2 (http/%77%77%77%2E%67%6F%6F%67%6C%65%2E%63%6F%6D)
This is an encoded version of the standard URL. The %77 value represents the HEX value 0x77. When this is converted to an ASCII code (character code 119), we get the lowercase letter w. %2E represents the Hex value 0x2E. When this is converted to an ASCII code (character code 46), we get the period or full stop character.
Example 3 (http://66.102.9.99)
This example shows the IP address of the Google web site. Entering this into the browser address box will take you directly to the Google page. The IP address can be further encoded.

Example 4 (http://%36%36%2E%31%30%32%2E%39%2E%39%39)
This is an encoded version of the IP address. As before, the %2E is translated to the Hex value 0x2E, which represents the ASCII code 46 which is a period or full stop. From the start of the encoded URL, %36 represents the Hex value 0x36. Once again, translate this into the ASCII character code and you will get the number 6. This represents the first digit of the IP address. Continue the translation and you will get all the digits of the IP address,

Example 5 (http://1113983331)
To decode this example you must take the decimal value above and convert it to Hexadecimal. The Hex value is 0x42660963. Split this into 42 66 09 63 and convert back to decimal as follows: 0x42 = 42, 0x66 = 102, 0x09 = 9 and 0x63 = 99. This shows the IP address.

Example 6 (http://%31%31%31%33%39%38%33%33%33%31)
This is an encoded version of the data from Example 5. %31 represents the Hex value 0x31, which in turn represents the ASCII code 49. The ASCII character for 49 is the number 1.

The Secure Sockets Layer

HTTP is not a secure protocol, and many networking protocols (like Ethernet) allow the conversation between two computers to be overheard by other computers on the same area of the network. The result is that it is very possible for a third party to eavesdrop on HTTP transactions and record authentication information, credit card numbers, and other important data.

Thus, Netscape developed the SSL (Secure Sockets Layer) protocol, which provides a secure communications channel that HTTP can operate across, while also providing security against eavesdropping and other privacy attacks. SSL has developed into an IETF standard and is now formally referred to as the TLS (Transport Layer Security) protocol (TLS 1.0 is essentially SSL 3.1). Not all browsers support TLS yet.

When your browser requests a URL that begins with https, it creates an SSL/TLS connection to the remote server and performs its HTTP transaction across this secure connection.

SSL works by using a private key to encrypt data that's transferred over the SSL connection. Both Netscape Navigator and Internet Explorer support SSL and many Web sites use the protocol to obtain confidential user information.

Another protocol for transmitting data securely over the World Wide Web is Secure HTTP (S-HTTP). Whereas SSL creates a secure connection between a client and a server, over which any amount of data can be sent securely, S-HTTP is designed to transmit individual messages securely. SSL and S-HTTP, therefore, can be
seen as complementary rather than competing technologies. Both protocols have been approved by the Internet Engineering Task Force (IETF) as a standard.

**The Request and Response Cycle**

When a web browser requests a web page, it sends a request message to a web server. The message always includes a header, and sometimes it also includes a body. The web server in turn replies with a reply message. This message also always includes a header and it usually contains a body.

There are two features that are important in understanding HTTP - it is a request/response protocol: each response is preceded by a request.

Although requests and responses each contain different information, the header/body structure is the same for both messages. The header contains meta-information (information about the message) and the body contains the content of the message.

For example, say you told your browser you wanted a document at http://localhost/index.html. The browser would connect to the machine at localhost on port 80 and send it the following message:

```
GET /index.html HTTP/1.1
Host: localhost
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/xbm, */*
Accept-Language: en
Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
```

Assuming that a web server is running and the path maps to a valid document, the server would reply with the following message:

```
HTTP/1.1 200 OK
Date: Sat, 18 Mar 2000 20:35:35 GMT
```
In this example, the request includes a header but no content. The response includes both a header and HTML content, separated by a blank line (see below).

```
<HTML>
<HEAD><TITLE>Sample Document</TITLE></HEAD>
<BODY>
  <H1>Sample Document</H1>
  <P>This is a sample HTML document!</P>
</BODY>
</HTML>
```

Figure 80

**HTTP Headers**

If you are familiar with the format of Internet email, this header and body syntax may look familiar to you. Historically, the format of HTTP messages is based upon many of the conventions used by Internet email, as established by MIME (Multipurpose Internet Mail Extensions). Do not be tricked into thinking that HTTP and MIME headers are the same. The similarity extends only to certain fields, and many early similarities have changed in later versions of HTTP.

Here are the important things to know about header syntax:

- The first line of the header has a unique format and special meaning. It is called a request line in requests and a status line in replies.
- The remainder of the header lines contain name-value pairs. The name and value are separated by a colon and any combination of spaces and/or tabs. These lines are called header fields.
- Some header fields may have multiple values. This can be represented by having multiple header fields containing the same field name and different values or by including all the values in the header field separated by a comma.
- Field names are not case-sensitive; e.g., Content-Type is the same as Content-type.
- Header fields don't have to appear in any special order.
Every line in the header must be terminated by a carriage return and line feed sequence, which is often abbreviated as CRLF and represented 0x0D 0x0A when seen in forensic software (little endian).

The header must be separated from the content by a blank line. In other words, the last header line must end with two CRLFs.

Browser Requests

Every HTTP interaction starts with a request from a client, typically a web browser. A user provides a URL to the browser by typing it in, clicking on a hyperlink, or selecting a bookmark and the browser fetches the corresponding document. To do that, it must create an HTTP request (see below).

Figure 81

In our previous example, a web browser generated the following request when it was asked to fetch the URL http://localhost/index.html

```
GET /index.html HTTP/1.1
Host: www.digital-detective.co.uk
User-Agent: Mozilla
```

From earlier in the chapter you know that the URL can be broken down into multiple elements. The browser creates a network connection by using the hostname and the port number (80 by default). The scheme (http) tells our web browser that it is using the HTTP protocol, so once the connection is established, it sends an HTTP request for the resource. The first line of an HTTP request is the request line, which includes a full virtual path and query string (if present); see below.

```
GET /index.html HTTP/1.1
Host: localhost
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/xbm, */*
Accept-Language: en
Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
```

From earlier in the chapter you know that the URL can be broken down into multiple elements. The browser creates a network connection by using the hostname and the port number (80 by default). The scheme (http) tells our web browser that it is using the HTTP protocol, so once the connection is established, it sends an HTTP request for the resource. The first line of an HTTP request is the request line, which includes a full virtual path and query string (if present); see below.

```
GET /index.html HTTP/1.1
Host: localhost
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/xbm, */*
Accept-Language: en
Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
```

The Request Line

The first line of an HTTP request includes the request method, a URL to the resource being requested, and the version string of the protocol. Request methods are case-sensitive and uppercase. There are several
request methods defined by HTTP although a web server may not make all of them available for each resource (see Table below). The version string is the name and version of the protocol separated by a slash. HTTP 1.0 and HTTP 1.1 are represented as HTTP/1.0 and HTTP/1.1. Note that https requests also produce one of these two HTTP protocol strings.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Asks the server for the given resource</td>
</tr>
<tr>
<td>HEAD</td>
<td>Used in the same cases that a GET is used but it only returns HTTP headers and no content</td>
</tr>
<tr>
<td>POST</td>
<td>Asks the server to modify information stored on the server</td>
</tr>
<tr>
<td>PUT</td>
<td>Asks the server to create or replace a resource on the server</td>
</tr>
<tr>
<td>DELETE</td>
<td>Asks the server to delete a resource on the server</td>
</tr>
<tr>
<td>CONNECT</td>
<td>Used to allow secure SSL connections to tunnel through HTTP connections</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>Asks the server to list the request methods available for the given resource</td>
</tr>
<tr>
<td>TRACE</td>
<td>Asks the server to echo back the request headers as it received them</td>
</tr>
</tbody>
</table>

Table 1

GET is the standard request method for retrieving a document via HTTP on the Web. When you click on a hyperlink, type a location into your browser, or click on a bookmark, the browser generally creates a GET request for the URL you requested. They should not alter information maintained on the web server; POST is intended for that purpose. GET requests do not have a content body.

If the browser has previously retrieved a resource, it may be stored within the browser's cache. In order for the browser to know whether to display the cached copy or whether to request a fresh copy, the browser can send a HEAD request. HEAD requests are formatted exactly like GET requests, and the server responds to it exactly like a GET request with one exception: it sends only the HTTP headers, it doesn't send the content. The browser can then check the meta-information (see section on ETags) contained in the headers, such as the modification date of the resource, to see if it has changed and whether it should replace the cached version with the newer version. HEAD requests do not have a content body either.

POST is used with HTML forms to submit information that alters data stored on the web server. POST requests always include a body containing the submitted information formatted like a query string. POST requests thus require additional headers specifying the length of the content and its format. These headers are described in the following section. Note that you may encounter browser warnings about expired pages if you attempt to revisit cached pages obtained via POST.
Request Header Field Lines

The client generally sends several header fields with its request. As mentioned earlier, these consist of a field name, a colon, some combination of spaces or tabs (although one space is most common), and a value (see below). These fields are used to pass additional information about the request or about the client, or to add conditions to the request.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type</td>
<td>text/html</td>
</tr>
</tbody>
</table>

Figure 83

A table of common HTTP request headers is shown below:

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>Specifies the target hostname</td>
</tr>
<tr>
<td>Content-Length</td>
<td>Specifies the length (in bytes) of the request content</td>
</tr>
<tr>
<td>Content-Type</td>
<td>Specifies the media type of the request</td>
</tr>
<tr>
<td>Authentication</td>
<td>Specifies the username and password of the user requesting the resource</td>
</tr>
<tr>
<td>User-Agent</td>
<td>Specifies the name, version, and platform of the client</td>
</tr>
<tr>
<td>Referrer</td>
<td>Specifies the URL that referred the user to the current resource</td>
</tr>
<tr>
<td>Cookie</td>
<td>Returns a name/value pair set by the server on a previous response</td>
</tr>
</tbody>
</table>

Table 2

Host

The Host field is new and is required in HTTP 1.1. The client sends the host name of the web server in this field. This may sound redundant, since the host should know its own identity; however, this is not always the case. A machine with one IP address may have multiple domain names mapped to it, such as www.bloggs.com and www.smith.com. When a request comes in, it looks at this header to determine what name the client is referring to it as, and thus maps the request to the correct content.

Content-Length

POST requests include a content body; in order for the web server to know how much data to read, it must declare the size of the body in bytes in the Content-Length field. Requests that do not have a content body, such as GET and HEAD, omit this field.

Content-Type

The Content-Type header must always be provided with requests containing a body. It specifies the media type of the message. The most common value of this data received from an HTML form via POST is application/x-www-form-urlencoded, although another option for form input (used when submitting files) is multipart/form-data.
Authorisation / Authentication

Web servers can require a login for access to some resources. If you have ever attempted to access a restricted area of a website and been prompted for a login and password, then you have encountered this form of HTTP authentication (see below). Note that the login prompt includes text identifying what you are logging in to; this is the realm. Resources that share the same login are part of the same realm. For most web servers, you assign resources to a realm by putting them in the same directory and configuring the web server to assign the directory a name for the realm along with authorisation requirements.

User-Agent

This field indicates what client the user is using to access the Web. The value is generally comprised of a name identifying the browser, its version number, and the operating system and platform on which it's running.

Here is an example from Netscape Communicator:

User-Agent: Mozilla/4.5 (Macintosh; I; PPC)

Microsoft Internet Explorer made the decision when it released its browser of also claiming to be “Mozilla,” which is Netscape's browser identification or name. Apparently this was done because a number of web sites used this field to distinguish Netscape browsers from others in order to take advantage of the additional features Netscape offered at the time. Microsoft made their browser compatible with many of these features and wanted its users to also take advantage of these enhanced web sites. Even now, the “Mozilla” moniker remains for the sake of backward-compatibility. Here is an example from Internet Explorer:

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)

Many web sites and web servers actually log browser User-Agent tags and store them in a database. This is why law enforcement personnel involved in CII (Covert Internet Investigations) need to be aware of the data their browsers are leaking, when making requests to web servers. I have seen some organisations actually brand their browser tags with information allowing them to be identified when a resource was requested.
**Referrer**

Unfortunately, the *Referrer* field was misspelled in the original protocol and, due to the need to maintain backward-compatibility; it is shown as *Referer*:

This field provides the URL of the last page the user visited, which is generally the page that linked the user to the requested page:

```
Referer: http://localhost/index.html
```

This field is not always sent to the server; browsers provide this field only when the user generates a request by following a hyperlink, submitting a form, etc. Unfortunately, browsers don't generally provide this field when the user manually enters a URL or selects a bookmark, since this apparently may involve a significant invasion of the user's privacy!

**Cookies**

Web browsers or servers may provide additional headers that are not part of the HTTP standard. The receiving application should ignore any headers it does not recognise. Examples of a pair of headers not specified in the HTTP protocol are *Set-Cookie* and *Cookie*, which Netscape introduced to support browser cookies.

*Set-Cookie* is sent by the server as part of a response:

```
Set-Cookie: cart_id=12345; path=/; expires=Sat, 18-Mar-05 19:06:19 GMT
```

This header contains data for the client to echo back in the *Cookie* header in future requests to that server:

```
Cookie: cart_id=12345
```

By assigning different values to each user, servers (and CGI scripts) can use cookies to differentiate between users.
Server Responses

Server responses, like client requests, always contain HTTP headers and an optional body. Here is the server response from our earlier example:

```
HTTP/1.1 200 OK
Date: Sat, 18 Mar 2000 20:35:35 GMT
Server: Apache/1.3.9 (Unix)
ETag: "74916-656-3562efde"
Content-Length: 141
Content-Type: text/html

<HTML>
<HEAD><TITLE>Sample Document</TITLE></HEAD>
<BODY>
  <H1>Sample Document</H1>
  <P>This is a sample HTML document!</P>
</BODY>
</HTML>
```

The structure of the headers for the response is the same as for requests. The first header line has a special meaning, and is referred to as the status line. The remaining lines are name-value header field lines. See below.

```
Status Line
HTTP/1.1 200 OK

Header Fields
Content-Type: text/html
Content-Length: 141
```

Figure 85

The Status Line

The first line of the header is the status line, which includes the protocol and version just as in HTTP requests, except that this information comes at the beginning instead of at the end. This string is followed by a space and the three-digit status code, as well as a text version of the status. See below.

```
HTTP/1.1 200 OK
```

Figure 86

Status Codes

Web servers can send any of dozens of status codes. For example, the server returns a status of 404 Not Found if a document doesn't exist and 301 Moved Permanently if a document is moved. Status codes are grouped into five different classes according to their first digit:

100-199

These status codes were introduced for HTTP 1.1 and used at a low level during HTTP transactions.
200-299
200-series status codes indicate that all is well with the request.

300-399
300-series status codes generally indicate some form of redirection. The request was valid, but the browser should find the content of its response elsewhere.

400-499
400-series status codes indicate that there was an error and the server is blaming the browser for doing something wrong.

500-599
500-series status codes also indicate there was an error, but in this case the server is admitting that it or a CGI script running on the server is the culprit.

Server Headers

After the status line, the server sends its HTTP headers. Some of these server headers are the same headers that browsers send with their requests. The common server headers are listed in the Table below.

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Base</td>
<td>Specifies the base URL for resolving all relative URLs within the document</td>
</tr>
<tr>
<td>Content-Length</td>
<td>Specifies the length (in bytes) of the body</td>
</tr>
<tr>
<td>Content-Type</td>
<td>Specifies the media type of the body</td>
</tr>
<tr>
<td>Date</td>
<td>Specifies the date and time when the response was sent</td>
</tr>
<tr>
<td>ETag</td>
<td>Specifies an entity tag for the requested resource</td>
</tr>
<tr>
<td>Last-Modified</td>
<td>Specifies the date and time when the requested resource was last modified</td>
</tr>
<tr>
<td>Location</td>
<td>Specifies the new location for the resource</td>
</tr>
<tr>
<td>Server</td>
<td>Specifies the name and version of the web server</td>
</tr>
<tr>
<td>Set-Cookie</td>
<td>Specifies a name-value pair that the browser should provide with future requests</td>
</tr>
<tr>
<td>WWW-Authenticate</td>
<td>Specifies the authorisation scheme and realm</td>
</tr>
</tbody>
</table>

Table 3

Content-Base
The Content-Base field contains a URL to use as the base for relative URLs in HTML documents. Using the `<BASE HREF=...>` tag in the head of the document accomplishes the same thing and is more common.

Content-Length
As with request headers, the Content-Length field in response headers contains the length of the body of the response. Browsers use this to detect an interrupted transaction or to tell the user the percentage of the download that is complete.
Content-Type
This field is provided with every response containing a body and must be included for all requests accompanied by a status code of 200. The most common value for this response is text/html, which is what is returned with HTML documents. Other examples are text/plain for text documents and application/PDF for Adobe PDF documents.

Date
HTTP 1.1 requires that servers send the Date header with all responses. It specifies the date and time the response is sent. Three different date formats are acceptable in HTTP:

Mon, 06 Aug 1999 19:01:42 GMT
Monday, 06-Aug-99 19:01:42 GMT
Mon Aug 6 19:01:42 1999

The HTTP specification recommends the first option, but all should be supported by HTTP applications.

ETag
The ETag header specifies an entity tag corresponding to the requested resource. Entity tags were added to HTTP 1.1 to address problems with caching. Although HTTP 1.1 does not specify any particular way for a server to generate an entity tag, they are analogous to a message digest or checksum for a file. Clients and proxies can assume that all copies of a resource with the same URL and same entity tag are identical. Thus, generating a HEAD request and checking the ETag header of the response is an effective way for a browser to determine whether a previously cached response needs to be fetched again.

Last-Modified
The Last-Modified header returns the date and time that the requested resource was last updated. This was intended to support caching, but it did not always work as well as hoped in HTTP 1.0, so the ETag header now supplements it. The Last-Modified header is restrictive because it implies that HTTP resources are static files, which is obviously not always the case.

Location
The Location header is used to inform a client that it should look elsewhere for the requested resource. The value should contain an absolute URL to the new location. This header should be accompanied by a 3xx series status code. Browsers generally fetch the resource from the new location automatically for the user. Responses with a Location field may also contain contents with instructions for the user since very old browsers may not respond to the Location field.

Server
The Server header provides the name and version of the application acting as the web server. The web server automatically generates this for standard responses.

Set-Cookie
The Set-Cookie header asks the browser to remember a name-value pair and send this data back on subsequent requests to this server. The server can specify how long the browser should remember the cookie and to what hosts or domains the browser should provide it.
WWW-Authenticate
As we discussed, web servers can restrict certain resources to users who provide a valid username and password. The WWW-Authenticate field is used along with a status code of 401 to indicate that the requested resource requires such a login. The value of this field should contain the form of authentication and the realm for which the authorisation applies. An authorisation realm generally maps to a certain directory on the web server, and a username and password pair should apply to all resources within a realm.

Proxies
Quite often, web browsers do not interact directly with web servers; instead they communicate via a proxy. HTTP proxies are often used to reduce network traffic, allow access through firewalls, provide content filtering, etc. Proxies have their own functionality that is defined by the HTTP standard. We don't need to understand these details, but we do need to recognise how they affect the HTTP request and response cycle. You can think of a proxy as a combination of a simplified client and a server (see below). An HTTP client connects to a proxy with a request; in this way, it acts like a server. The proxy forwards the request to a web server and retrieves the appropriate response; in this way, it acts like a client. Finally, it fulfils its server role by returning the response to the client.

The picture below shows how an HTTP proxy affects the request and response cycle. Note that although there is only one proxy represented here, it's quite possible for a single HTTP transaction to pass through many proxies.

Proxies affect us in two ways. First, they make it impossible for web servers to reliably identify the browser. Second, proxies often cache content. When a client makes a request, proxies may return a previously cached response without contacting the target web server.

Figure 87
Caching

One of the benefits of proxies is that they make HTTP transactions more efficient by sharing some of the work the web server typically does. Proxies accomplish this by caching requests and responses. When a proxy receives a request, it checks its cache for a similar, previous request. If it finds this, and if the response is not stale (out of date), then it returns this cached response to the client. The proxy determines whether a response is stale by looking at HTTP headers of the cached response, by sending a HEAD request to the target web server to retrieve new headers to compare against, and via its own algorithms. Regardless of how it determines this, if the proxy does not need to fetch a new, full response from the target web server, the proxy reduces the load on the server and reduces network traffic between the server and itself. This can also make the transaction much faster for the user.

Because most resources on the Internet are static HTML pages and images that do not often change, caching is very helpful. For dynamic content, however, caching can cause problems. CGI scripts allow us to generate dynamic content; a request to one CGI script can generate a variety of responses. Imagine a simple CGI script that returns the current time. The request for this CGI script looks the same each time it is called, but the response should be different each time. If a proxy caches the response from this CGI script and returns it for future requests, the user would get an old copy of the page with the wrong time.

Fortunately, there are ways to indicate that the response from a web server should not be cached. We'll explore this shortly. HTTP 1.1 also added specific guidelines for proxies that solved a number of problems with earlier proxies. Most current proxies, even if they do not fully implement HTTP 1.1, have adopted these guidelines.

Caching is not unique to proxies. Browsers do their own caching as well. Some web pages have instructions telling users to clear their web browser's cache if they are having problems receiving up-to-date information. Proxies present a challenge because users cannot clear the cache of intermediate proxies (they often may not even know they are using a proxy) as they can for their browser.

Encoding

A server may be able to speed up the download of a large document to this client by sending an encoded version of the document. The browser should decode the document automatically for the user. Options for encoding include GZIP, compress, and identity (no encoding).

Here is an example header specifying that the browser supports compress and GZIP:

```
Accept-Encoding: compress, gzip
```

The problem this presents to us when conducting the forensic analysis of internet cache, is that some of the data may be compressed or encoded.
Redirects

How Redirects Work

Redirects provide a way to transport the browser from one point on a web site to another. Redirects are most commonly used to translate references to outdated web pages to new, updated ones. They can be instigated either by the browser, in which case they are referred to as client-side redirect, or by the server, in which case they are referred to as server-side redirects.

Virtually all webmasters at some point will discover that not all links to their site end up where they were intended. For example, if the author of another web site places a link to your site, but misspells the hyperlink, the result will be a 404 error every time that link is selected.

Another common occurrence is site reorganisations. As you move files around, rename them or even delete them entirely, you will find visitors will receive 404 errors. This is because other sites, search engines and directories often link to those pages. It is an impossible task to be able to change all the hyperlinks to your site.

If the webmaster determines that someone has linked to a non-existent page, there are ways of redirecting the user to the correct page.

Browser / Client Side Redirects

So how does a webmaster recapture the traffic to these lost pages? There are several things that can be done.

Most webmasters use a special Meta tag called "refresh" to direct their visitors to the new page. To do this they create a blank page which contains this tag (it is placed in the header). The tag simply says "wait a few seconds then go to a named page".

```html
<META HTTP-EQUIV="refresh" content="1; URL=http://www.digital-detective.co.uk/index.asp">
```

This example waits one second and then loads the specified page into the browser.

Normally, the webmaster will wait for around 10 seconds before asking the browser to go to the new page. This is so the user can click on a hyperlink to the new page, if for some reason the browser does not go to the new URL.

You should note that according to the W3C organisation (one of the web's top standards groups):

"Some user agents support the use of META to refresh the current page after a specified number of seconds, with the option of replacing it by a different URL. Authors should NOT use this technique to forward users to
different pages, as this makes the page inaccessible to some users. Instead, automatic page forwarding should be done using server-side redirects.

**Server-Side Redirects**

There are many ways to implement server-side redirects. One of the most common is to use the `.htaccess` file, which is supported by the Apache web server. `.htaccess` supports a directive called redirect. This directive transparently changes the URL to a new URL. However, this is not the usual method for implementing Server-Side redirects as not all hosts support this.

There are other server-side redirection tricks. You can use special ASP, PHP or even CGI commands to move the user to a new page. The following example is an ASP implementation.

```asp
<%
' Begin buffering the HTML
' Note this MUST happen before the initial <HTML> tag.
Response.Buffer = True
%>
<HTML>
<BODY>
<p>HTML code before potential redirect.</p>
<%
' The following lines actually perform a redirect to the Google site
If 1 = 1 Then
  Response.Clear
  Response.Redirect 'http://www.google.com'
End If
%>
```

**Redirects and NetAnalysis**

Embedded within the CACHE INDEX.DAT file are numerous internet history records that have REDR as the record header. This header is a REDIRECT entry and is evidence of a SERVER-SIDE redirect. Client Side redirects are NOT recorded within the INDEX.DAT files. The REDR entry is a page that has been visited, and has caused a SERVER-SIDE REDIRECT. This entry reflects the page that has caused the redirection and will NOT show where the server redirected the visitor.

**Redirect**

Unfortunately, SERVER-SIDE redirects recorded in the cache INDEX.DAT do not have date/time values associated with them. The binary INDEX.DAT file is not always written in a sequential manner, therefore it is very difficult to identify which page or site the web server has redirected the browser. It may be possible to look at the lower level domain (host) of the redirected entry to try and identify which site has been visited as a result of the redirection.
Redirection is nothing unusual or sinister! It is a legitimate tool for webmasters to prevent pages not being found. If a webmaster were to redirect a visitor to an unwelcome page, then he/she would have the choice not to view that site any further. This type of defence can be negated by identifying web activity which shows deliberate action on behalf of the user. Examination of search engine criteria stored in history files or data located in protected storage in the registry may assist the examiner.

Forensic examiners may well find redirect entries that contain search criteria. Many search engines will employ this method to pass search criteria to scripts to deliver advertising material based on the search criteria. For example, if the suspect searched for ‘mobile telephones’, the server may display in the search result pages, advertising which relates in some way to mobile telephones.

Even though the redirect entry may not show a date/time value, it is evidence of a page being visited. The key point to remember is that this is the URL that caused the redirection and not the page to which the browser was redirected.
Understanding Date & Times

Date / Time Values in Binary Files

When analysing Internet History, it is very important that you understand the significance of the Secondary Date and how it works within different history files. It is also important to know how the date/time values are stored within the different history/cache files and how to decode them manually if necessary.

Internet Explorer

Microsoft Internet Explorer uses a date/time value called a FILETIME. FILETIMES are 64bit values (stored in 8 bytes - little endian). A typical URL record entry with the header shown and date values outlined in red. This is an entry from a CACHE INDEX.DAT file. There are two FILETIMES available, located at offset decimal 8 and 16 from the start of a URL record.

Figure 88

The FILETIME structure holds an unsigned 64-bit date and time value. This value represents the number of 100-nanosecond units since the beginning of January 1, 1601 UTC.

```
typedef struct _FILETIME{
    DWORD dwLowDateTime;
    DWORD dwHighDateTime;
} FILETIME, *PFILETIME, *LPFILETIME;
```

- dwLowDateTime
  - Low-order 32 bits of the file time value.
- dwHighDateTime
  - High-order 32 bits of the file time value.
Coordinated Universal Time (UTC)

The times of various events, particularly astronomical and weather phenomena, are often given in "Universal Time" (abbreviated UT) which is sometimes referred to, now colloquially, as “Greenwich Mean Time” (abbreviated GMT). The two terms are often used loosely to refer to time kept on the Greenwich meridian (longitude zero), five hours ahead of Eastern Standard Time. Times given in UT are almost always given in terms of a 24-hour clock. Thus, 14:42 (often written simply 1442) is 2:42 p.m. Sometimes the character Z is appended to a time to indicate UT, as in 0935Z (Zulu).

When a precision of one second or better is needed, however, it is necessary to be more specific about the exact meaning of UT. For that purpose different designations of Universal Time have been adopted. In astronomical and navigational usage, UT often refers to a specific time called UT1, which is a measure of the rotation angle of the Earth as observed astronomically. It is affected by small variations in the rotation of the Earth, and can differ slightly from the civil time on the Greenwich meridian.

However, in the most common civil usage, UT refers to a time scale called "Coordinated Universal Time" (abbreviated UTC), which is the basis for the worldwide system of civil time.

During the winter months, UTC is the civil time scale for the United Kingdom and Ireland. It is also equivalent to the civil time for Iceland, Liberia, Morocco, Senegal, Ghana, Mali, Mauritania, and several other countries.

One can think of UT1 as being a time determined by the rotation of the Earth, over which we have no control, whereas UTC is a human invention. It is relatively easy to manufacture highly precise clocks that keep UTC, while the only "clock" keeping UT1 precisely is the Earth itself. Nevertheless, it is desirable that our civil time scale not be very different from the Earth's time, so, by international agreement, UTC is not permitted to differ from UT1 by more than 0.9 second. When it appears that the difference between the two kinds of time may approach this limit, a one-second change called a "leap second" is introduced into UTC. This occurs on average about once every year to a year and a half.

Greenwich Mean Time is a widely used historical term, however, due to ambiguity, its use is no longer recommended in technical contexts.

Extracting Raw Hex Date Time Values

If we extract the RAW hex, the examples shown in Figure 88, we can calculate the date as follows:

```
00 68 FC 64 C8 43 c4 01
0x01C443C864FC6800 at offset 0x08 of the URL Record
Equates to Thu, 27 May 2004 08:55:44 UTC
```

Figure 89 shows the DCode Date software (available from http://www.digital-detector.co.uk) with the HEX values already entered. When decode is pressed, the true UTC date/time value is returned.
Once again, the second date/time hex value can be extracted and decoded with the software. Select the 8 bytes from offset 16 (0x10) of the URL record and enter them as shown in Figure 90.

```
D0 27 A7 76 65 C4 01
0x01C45576E7A727D0 at offset 0x10 of the URL record
Equates to Fri, 18 Jun 2004 20:57:45 UTC
```

Figure 90

Each Internet History Record has the possibility of having 2 x 64bit date/time values. NetAnalysis refers to these dates as the Last Visited date and the Secondary date. Unfortunately, within each INDEX.DAT file, the dates are stored in different ways and put to different uses. Each INDEX.DAT type is detailed below with the meaning of the date/time value and how NetAnalysis presents them to the examiner.

**Master History Records**

The Master History INDEX.DAT file is covered in more detail on Page 29. The role of the MASTER INDEX.DAT file is to record the web addresses that have been visited along with the Last Visited date/time and visit count. The forensic investigator can identify a MASTER history record within NetAnalysis by looking at the “Index Type” column for the value “History”. This value is also mirrored in the status bar. The date/time data is as follows:

```
INDEX.DAT Raw Data
Offset 0x08 - Last Visited Date / Time in UTC
Offset 0x10 - Last Visited Date / Time in UTC

NetAnalysis Representation
Last Visited - Last Visited Date / Time converted to the Time Zone Standard Time *
Secondary Date - Last Visited Date / Time UTC
```
The raw data in the INDEX.DAT file should be identical. NetAnalysis will convert the first date to the local time zone as set in NetAnalysis Properties. You must set your Local Time Zone prior to extracting the data. If the Time Zone settings are incorrect, you will be presented with misleading date/time values. This is explained further on page 12. The default Time Zone for NetAnalysis is UTC, but this can be changed by the investigator.

**Daily History Records**

Internet Explorer maintains a Daily record of visited pages. This INDEX.DAT file has a unique HOST record entry. The HOST record entry is used by Internet Explorer to display the hierarchical history structure when showing which web sites have been visited. This file also has two 64 bit date/time values. Unlike the Master History file, the Daily file stores every visit to a web site on a particular day. Unfortunately, these records get deleted on a regular basis as the data is converted on a weekly basis into the Weekly file. When examining a Daily file the date/time values are as follows:

---

**INDEX.DAT Raw Data**

Offset 0x08 – Last Visited Date / LOCAL TIME
Offset 0x10 – Last Visited Date / UTC

**NetAnalysis Representation**

Last Visited – Last Visited Date / Time converted to the Time Zone Standard Time *
Secondary Date – Last Visited Date / LOCAL TIME

This particular file type is very important as you can establish the time zone setting offset from UTC at the time of browsing. This information is also checked and recorded within the system log which is available from the Reports menu. Figure 91 shows the system log with the verified TZ setting.
If the Last Visited and Secondary date is identical, then you know there is no difference between the primary and secondary date, therefore the time zone offset was zero. This means that in this case, the time zone setting was UTC (assuming NetAnalysis TZ Settings are on UTC to start).

Your Time Zone setting must be set to standard time in NetAnalysis for the Last Visited dates to reflect Standard Time. For forensic work, it is good practice to have date and times calculated from Standard Time.

Weekly History Records

At the end of a browsing week, all the daily INDEX.DAT file content is archived into a weekly INDEX.DAT file. The actual date details within the binary file changes for this file type. When the weekly INDEX.DAT file is created, this created date is saved at offset 0x10 of every URL record. There is not much evidential value to using this date, so NetAnalysis discards it. If you need to verify this date/time when compared to the file date/time stamp, you can do this manually using your forensic tool and DCode.

NetAnalysis takes the local date/time value at 0x08 and converts it to UTC, taking into account daylight saving information. In then adjusts it depending on the TZ offset manually set in NetAnalysis prior to extraction. The local time stored at 0x08 is placed in the secondary date/time column so the accuracy of your TZ adjustments can be verified. No TZ translation is added to this date/time value.

<table>
<thead>
<tr>
<th>INDEX.DAT Raw Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset 0x08 - Last Visited Date / LOCAL TIME</td>
</tr>
<tr>
<td>Offset 0x10 - Created Date of INDEX.DAT file UTC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NetAnalysis Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last Visited - Last Visited Date / Time converted to the Time Zone *</td>
</tr>
<tr>
<td>Secondary Date - Last Visited Date / LOCAL TIME</td>
</tr>
</tbody>
</table>
Cache (Content) Records

This INDEX.DAT file controls the data saved within the cache. It plays a very important part in the browsing process. This file type can be found within the \Content.ie5 folder. This is the index to the cache. Every time a user visits a page, the files that make up the page may be JPEGs, GIFs, PNGs, flash movies etc. The files themselves are saved into the Temporary Internet Files folder. The files will be renamed slightly and listed within the INDEX.DAT file.

**NOTE:** Other applications also use the cache for temporary storage without writing data to the INDEX.DAT files. For example, Outlook Express uses the cache to render E-Mail messages and decoding Base64 attachments. Those files will have no INDEX.DAT entry and are saved as temporary files. They also do not conform to the standard cache naming convention. Outlook Express cache naming convention is wbk***.tmp where *** represents a 3 or 2 digit hex value. E.g. wbke2.tmp

If the user returns to a page that has been visited previously, Internet Explorer can extract the data from cache rather than have to download the item again. This speeds up the browsing process. However, Internet Explorer needs to know the last time these files were updated so that it can download a fresh copy if the file has been updated. To allow it do this, it records the Last Modified date and time of every file that makes up a web page. This data is kept in the secondary date / time field within NetAnalysis and is stored as a UTC Value. It is NOT translated by NetAnalysis TZ Settings.

**Note:** The last modified date/time is associated with the resource on the Web Server – it is NOT set by or recorded by the user’s machine. Some Webmasters use this date/time stamp for version control.

<table>
<thead>
<tr>
<th>INDEX.DAT Raw Data</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset 0x08 – Last Modified Date/Time of resource in UTC</td>
<td></td>
</tr>
<tr>
<td>Offset 0x10 – Last Visited Date/Time in UTC</td>
<td></td>
</tr>
</tbody>
</table>

| NetAnalysis Representation |  |
| Last Visited – Last Visited Date / Time converted to the Time Zone * |  |
| Secondary Date – Last Modified Date/Time in UTC |  |

This data is extracted straight from the file on the web server (the date returned is in GMT) and makes no difference whether the clock on the suspect system is accurate or not. For page content where the site is updated regularly such as news information sites, this is a good indicator as to how accurate the system clock was at the time of visit. This is only a guide though and should not be relied upon in evidence. To verify the actual last visited date / time on the web server, simply right click on the URL and select Get Web Server - Last Modified Date.

For example, HOTMAIL pages are server generated and contain a generation date/time stamp. If the examiner searches for ct=, they will find a 10 digit number which represents a UNIX cTime (number of seconds from 1\(^{st}\) January 1970 00:00:00) which then can be easily converted into a date/time using DCode.
NetAnalysis Support & Purchase

United Kingdom / Europe / Rest of World

For registered users of NetAnalysis, there is a technical support forum available at the digital-detective web site. Please carefully follow the instructions when applying for the forum. Once you have created a username and received your password, you must complete the web based form to request access to the forums, including the NetAnalysis support forum. Access to the support forum will only be allowed when a valid Licence ID is provided.

http://www.digital-detective.co.uk/cgi-bin/digitalboard/YaBB.pl

For UK sales please visit the following link:

http://www.digital-detective.co.uk/purchase.asp

Technical support will only be provided via the online forum. Please also remember the Time Zone factor. I am in the UK and normally deal with technical support issues between 1800 and 2200 hours GMT.
References

- Astronomical Applications Department of the U.S. Naval Observatory (http://aa.usno.navy.mil/faq/docs/UT.html)
- Wikipedia (http://en.wikipedia.org)
- The Microsoft Developer Network (http://msdn.microsoft.com/)
- CGI Programming with PERL By Scott Guelich, Shishir Gundavaram and Gunther Birznieks
- Internet RFC/STD/FYI/BCP Archives (http://www.faqs.org/rfcs/)
- Internet Assigned Numbers Authority (http://www.iana.org/)
- World Wide Web Consortium (http://www.w3.org/)
- Netscape (http://www.netscape.com/)
- Mozilla (http://www.mozilla.org/)
- Microsoft Internet Explorer (http://www.microsoft.com/windows/ie/default.mspx)
- Microsoft TechNet (http://www.microsoft.com/windows/ie/default.mspx)
- Digital Detective (http://www.digital-detective.co.uk)
- Gemal.dk - Gemal's psyched site (http://gemal.dk/)
- Microsoft Windows XP Registry Guide
- Microsoft Internet Explorer 6 Resource Kit
Special Thanks

With Thanks to the Following People

Writing and supporting software is a mammoth task. The creation of NetAnalysis has taken a considerable amount of time over the past few years. Working on such a project would not be possible if it were not for the support of friends and colleagues.

I would like to take this opportunity to thank everyone who has supported me over the past few years, in particular all the beta testers on Digital Detective.