Automating Web History Analysis

Michael Sonntag

Institute for Information Processing and Microprocessor Technology (FIM)

Johannes Kepler University Linz

sonntag@fim.uni-linz.ac.at

Abstract

Web browsing becomes more important as more and more applications are moved towards the WWW. Consequently traces of such web activity are increasing in importance for law enforcement or companies. But the web browser market is getting more fragmented, as more browsers are being used, and even a single person might employ different browsers on several devices, like IE and Firefox on a PC and another one on a mobile device (PDA/Pad). This paper lists the information which can be found for the three most important browsers (IE, Firefox, Chrome) and presents a tool for collecting and collating web history information from a forensically acquired media for all users and all browsers in a forensically sound way. This paper additionally analyzes requirements for a more encompassing analysis framework.

1. Introduction

Computer forensics is an important area of investigating criminal or undesirable activity. In many cases a very prominent part (often the most important one) is browser forensics, i.e. investigating the activity of the suspect on the computer regarding web browsing. This consists of identifying the behaviour in as many details as possible regarding, for example:

- Which web pages were visited
- What content existed on these pages and whether this content is still available (on the computer – in the web it might already have changed or been removed!)
- Was the user there intentionally or was it a popup, dynamically loaded image etc.
- Can we find hints to external services (Webmail services, file storage, ...)
- What additional information is available (e.g. visit count, usernames/passwords)

This is difficult in practice, as previously there were only few browsers (e.g. predominantly Internet Explorer 6), but their number has grown over the last years and several browsers are currently reaching large shares of the audience. This number is potentially growing, as PDAs and Pads again use different browsers, or at least different versions or configurations of them. Another difficulty is that many persons use different browsers (e.g. IE for business applications requiring this one, Firefox for “normal” websites, and Webkit on their Android Smartphone). To obtain a complete picture of the web-related activities, the individual results therefore need to be combined. This leads to difficult and time-consuming gathering of data from different sources and various formats with the subsequent need for collating the results, which is exacerbated by the fact that most tools exist for some browsers only and produce output in their own special format. Additionally many common (rather simplistic and not suited for forensics) programs only allow inspection via a graphical user

interface of a limited set of data, but no export for further handling and contain no provisions to ensure the chain of custody or at least data integrity.

What is needed is a tool to automate collecting data from various sources and integrating them. As computer forensic is still an area for experts and often needs automation, it should offer a command line interface for unattended operation or scripting. Ideally the software should also be independent of the actual web browsers (e.g. Firefox lets you access the cache through the URL “about:cache”), so it can be used on images of the file system. A first (and currently limited to the web history, cookies and cache) example of such a software has been implemented and is presented in this paper. Additionally it describes and compares the information present in several important web browsers.

2. Data stored by the different browsers

A lot of data is potentially available in a browser, although not everything is the same for all browsers. Therefore it is important to identify a “baseline” of information which will exist for all of them. This information can then be compared and is the minimum to be expected for evaluation. Additional information might exist, if a specific browser has been employed, but cannot necessarily be expected. This investigation is limited to the most important browsers at the moment: Internet Explorer, Firefox and Chrome. Additionally only recent versions of them are described (several years old ones might contain different data, employ a different format or use different locations for storing it).

One additional difficulty exists: most (or all) of this data is not stored if the browser is used in a “private” mode (”Private Mode”, “InPrivate Browsing”, ...). Even then traces might remain on the disk (e.g. Internet Explorer does store some data on the disk, but deletes it on leaving this mode), especially in the paging file, which contains copies of parts of the RAM. However, there is neither a guarantee for this, nor is this a complete or consistent view and it might also contain traces from very old activities, other users, and browsing in “normal” mode. This is therefore excluded and remains a task for manual recovery through experts.

What is not investigated here too, are extensions: A large number exists in different formats, depending on the browser. These may store any kind of additional data like searches, download histories, page ratings etc. A manual inspection which plugins/extensions are present is therefore a necessity. Only for some very common ones would integration into an automatic system be worthwhile.

2.1. Common aspects

When URLs are stored, they are typically full URLs, meaning they include any parameters which might be present. These can be very interesting, e.g. when using search engines. All major ones do not use the POST method (search words would be encoded in the HTTP content part), but rather the GET method, where search words are encoded into the URL. This might be even more interesting if auto-completion is activated, as then a multitude of requests will exist in the history, showing the typing sequence, including any searches not actually executed.

2.2. Internet Explorer

Internet Explorer forensics is problematic, as most data is stored only in binary files in an undocumented format. Most of the data has been decoded in the meantime [5, 6], so it can be extracted. For many elements separate tools exist, where unification is a necessity for analysis. It must also be noted, that the hit count (for URLs, cookies etc.) is not necessarily accurate in Internet Explorer [1]. Regarding the various most interesting elements, the following data fields can be retrieved:

- Bookmarks: URL and icon-URL are stored in the bookmark file. The file's creation date shows the date and time when this bookmark was stored. As they cannot be created without user intervention, they show intentionality.
- History: Last modified time and last access time, URL, hit count (index.dat[1]). A very important part (see common elements) for forensic analysis. Potentially problematic is, that the history does not show web pages but all elements (images, stylesheets etc.), and that there is no guarantee for intentional access to them. All embedded elements are loaded automatically (→ history entry) and even web pages can be "added" to it without user intervention (iframes, moving to a page or opening a popup through JavaScript).
- Typed URLs: Manually typed URLs are stored in the registry. As a timestamp is available only for the whole key (but not the individual values), only one time of modification exists, which is the time the last URL was typed. The list is limited to the last 25 and Internet Explorer stores only those URLs, which are fully manually typed. If a URL is begun but automatically completed, it will not be stored here [2].
- Cache: File creation time (Stored in the file system, allows identifying the first visits), last modified/last access time (date and time of last change of the content of the file which was seen and the last time this element was displayed), hit count (how often this element was show; see above for reliability), HTTP headers (typically not very interesting), URL and filename (index.dat) and file content (file system, separated into cache directories). In practice the cache is today not that useful anymore, as it will contain only files where caching was not prohibited by the server. This is typically done for dynamically generated pages (which change often and must be reloaded on every visit) or because of security, like webmail, where all static content (advertisements, menu bars etc.) will be cached, but the actual message content not.
- Cookies: Name and Value (as set by the server), the domain/path this cookie will be sent to, expiration and creation time, whether to send it across unencrypted connections (all from the cookie file), last modified time (index.dat: the last date and time this cookie was set, not necessarily the last time it was sent to the server!), hit count (not very useful for cookies!)
- Form completion data and stored passwords: Name/value respectively username/password are stored in several places. However, in recent versions they are mostly inaccessible without knowledge of the URL, as this is used as their encryption key. Therefore it must be known, retrieved from the history etc. [3, 4, 12] for accessing the data.
- Certificates: Certificate content and private keys (encrypted) are stored system-wide and are typically of very little interest. Only if client certificates are used or additional server or CA certificates have been installed these could potentially be useful.

---

[1] The file "index.dat" is a special file type used for several elements and existing in multiple locations.

2.3. Firefox

Firefox used in early versions are very strange and complex file format, but since version 3 the data is quite easily accessible as several SQLite databases [7].

- Bookmarks: URL, title, date added/last modified as well as its location (menu bar, folders etc.) are stored in places.sqlite (table moz_bookmarks). Additional information is available in combination with the table moz_places, identical as for history entries (see below). Forensically this data is very important. Some additional information, last bookmark folder used, bookmark descriptions, date of addition and of last modification, is in table moz_items_annos.
- History: Common elements include URL, title, visit count and date/time of the last visit (places.sqlite in table moz_places) [8, 9]. Additional information is whether such a visit was "hidden", i.e. through a RSS, iframes or Javascript calls. This can be useful to confirm (a denial) of an intentional visit. The field "typed" specifies, whether the user typed at least a portion (autocomplete; more information in the table moz_inputhistory on what was actually typed) of the URL manually. The field "frequency" is a measure (not count) of how often a URL was visited and how recent the last visit was. It is used to determine the ordering of suggestions for autocompletion. The actual sequence of visits is stored in moz_historyvists where for each visit the preceding "location" is listed together with a timestamp of the visit, a session identifier (allowing reconstruction of visits belonging together) and a visit type. The latter describes how the user "arrived" as this visit: by following a link, typing an URL (possible in combination with autocompletion or through selecting a history entry) or using a bookmark. This includes separate values for downloads, permanent and temporary redirections and embedded content (images, iframes and frame content) too.
- Cache: Complete request string (URL), file size, expiration/first access/last access timestamp, access count and the HTTP response header. The file data itself is stored in three different archives (small/medium size files) or directly on the disk (files larger than 12 kB) identically as received from the server without any changes, but under a random name to be found alongside the other metadata [10, 11].
- Cookies: Name/value pairs of the cookie content, the host/path they will be sent to, which connections to send them over (encrypted only, ...) as well as expiry, last access and creation timestamps are present (cookies.sqlite).
- Form completion data and search history: Field name and the stored value, how often this information was used, date/time of first and last use (formhistory.sqlite). If present, this is very useful information as it contains content as well as timestamps. Under the name "searchbar-history" the information from the web search field is stored. Note that this information may be encrypted with a master password (similar to stored passwords).
- Stored passwords: Password for sites are stored in signons.sqlite in combination with key3.db (master key for decrypting saved password). Information contained is the username and the password, hostname, how often it was used, form submission URL, relevant field names as well as timestamps for creation, last use and last password change. This is extremely important as users often reuse usernames and password for different sites and services. Also the timestamps allow retrieving a lot of information regarding web site visits.
- Certificates: Specially accepted certificates (certificate exceptions; could not be verified as valid, but the user confirmed these) are stored in the file cert_override.txt. These signify that the user has visited a site presenting these certificates at least once and intentionally confirmed

to go there. However, this does not prove that the user visited a specific site: they might have been presented by any site, especially as they were seen as untrusted by the default algorithms. The certificates themselves are stored in cert8.db.

- Download history: The source and destination of downloaded files (unless cleared) is found in downloads.sqlite. This additionally includes start- and end time, referer (web page from which the download was initiated, file size, mime type and state (and a few other less important elements like preferred applications for opening). Additional info is contained in places.sqlite (moz_annos and moz_anno_attributes) regarding the filename of downloads.
- Stored sessions: This allows restoring the session state, e.g. when closing the browser with several tabs open or after a crash (sessionstore.js). Forensically this is typically not very useful.
- DOM storage: Allows web pages to permanently store data similar to cookies as name/value pairs under the same-origin policy (webappstore.sqlite). Forensically this is potentially very interesting as a lot of data may be stored, but because of infrequent use by web pages this seems to be practically of less importance.
- Website-specific data: Firefox stores a few preferences for sites, namely up- and download directory as well as whether the content should be shown in full zoom or not (content-prefs.sqlite). The individual settings are not that interesting, but the groups show often visited sites, which may be useful as corroboration and as proof that some content was downloaded. Additional information can be found in permissions.sqlite regarding allowing popups, installing extensions, setting cookies and showing images. Within places.sqlite (moz_favicons) is information about the website icons, including its URL (requested only for site visits, therefore useful) and an expiration timestamp.

2.4. Chrome

<To be completed>

3. Implementation

A very important aspect of computer forensics is authenticity and integrity of evidence. It is therefore necessary to include hash values for all source and result files and work on forensic copies of media, typically disk images. These and other necessities led to the design and implementation of a tool to extract browsing data from multiple sources within files from several browsers and compile them to a single collection. This is especially important, as e.g. IE deletes data from the cookie, cache and history storage separately, so to achieve the most comprehensive view all elements must be combined.

3.1. Existing tools

Some tools already exist to help in the evaluation of browser data. The most important ones are:

- Web historian[2]: A windows tool to extract the web, cookie, download and form history from IE, Firefox, Chrome and Safari. Apart from the GUI output is possible in HTML, XML or CSV. Working on disk images is only possible through manual mounting and special configuration, which is a significant drawback. Some analysis functions (website frequency by domain, daily timeline etc) are available. It supports setting a UTC offset to account for timezones. No command line interface is available, so it is unsuited for automation.
- Historian[3]: This Windows tool supports extracting cookies, downloads, bookmarks, history, form data and cache from Chrome, IE, Firefox and Opera (not everything from each browser, however). It creates hash values of the source files in the output. Export is possible as text and CSV, but can be customized through templates. This is purely a "converter", i.e. there is no possibility of inspecting the data through the GUI, and it can be used through the command line.
- Pasco and galleta[4]: Open source tools for parsing Internet Explorer index.dat, respectively cookie files. Very useful for these purposes, but no other functionality at all. They are available in source code and run on Linux too.

Several other tools are available, but these are predominantly Windows-only tools, merely support small areas (typically cache viewing only) and have little, if any, export functionality. These, as well as those described above, therefore suffer from some shortcomings, necessitating a new tool.

3.2. System description

This command line tool extracts the history, cache and cookies, and produces a timeline, and groups individual requests to sessions. The output is possible as text, CSV, XML or a SQLite database. As the tools extracts all the information from the disk image, the results might be very large, e.g. on a server with numerous users. Therefore it can be restricted to certain users or browsers. If desired, it employs an external tool[5] to try to reconstruct deleted files which might contain additional information to be extracted. As a special feature it is possible to compare the cache against the current state of the webserver, i.e. the tool will retrieve the found URLs and compare the webpage/image/... against the data found in the cache, in this way ascertaining whether the resource is still available and whether it has changed in the meantime (or some manipulation has taken place – which is the case remains for the examiner to find out).

The tool was implemented as a set of bash scripts employing only common external programs and can therefore be run on any kind of Linux, and even under cygwin in Windows (with the exception that write-protection of input files cannot be enforced and only NTFS and FAT are supported as file systems to be investigated, as opposed to all locally mountable file systems under Linux).

---

[2] Mandiant, Web Historian, http://www.mandiant.com/products/free_software/web_historian/ (20.4.2012)

[3] GaiJin: Historian, http://www.gaijin.at/dlhistorian.php (20.4.2012)

[4] Pasco, http://www.mcafee.com/us/downloads/free-tools/pasco.aspx (20.4.2012), Galleta, http://www.mcafee.com/us/downloads/free-tools/galleta.aspx (20.4.2012)

[5] CGSecurity: PhotoRec, http://www.cgsecurity.org/wiki/PhotoRec (20.4.2012)

3.3. Data extraction process

<Description of how the tool works with a special focus on how this can be generalized>

3.4. Exemplary scenario

<Exemplary scenario (description of what the used did) and screenshots/output of the tool>

4. Conclusions

<To be completed>

5. Literature

[1] Paul Andrews, Hit Counter Accuracy – Caveat Emptor! http://www.browserforensics.com/?p=155 (20.4.2012)

[2] Lih Wern Wong, Forensic Analysis of the Windows Registry, http://www.forensicfocus.com/downloads/forensic-analysis-windows-registry.pdf (20.4.2012)

[3] Nir Sofer, IE PassView, http://www.nirsoft.net/utils/internet_explorer_password.html (20.4.2012)

[4] Kunal Sachdeva, How To Extract Saved Password From Internet Explorer, Mozilla Firefox, Google Chrome, Yahoo Messenger, Gtalk And Msn http://kunalsachdeva.wordpress.com/2009/02/26/how-to-extract-saved-password-from-internet-explorer-mozilla-firefox-google-chrome-yahoo-messenger-gtalk-and-msn/ (20.4.2012)

[5] Keith Jones, Forensic Analysis of Internet Explorer Activity Files, http://nys.fd.org/cja/forensics/ieactivity.pdf (20.4.2012)

[6] Profile folder – Firefox, http://kb.mozillazine.org/Profile_folder_-_Firefox (20.4.2012)

[7] David Koepi, Firefox Forensics, http://davidkoepi.wordpress.com/2010/11/27/firefoxforensics/ (20.4.2012)

[8] Kristinn Guðjónsson, Firefox 3 History, http://computer-forensics.sans.org/blog/2009/07/15/firefox-3-history/ (20.4.2012)

[9] Mozilla Firefox 3 History File Format, http://www.forensicswiki.org/wiki/Mozilla_Firefox_3_History_File_Format (20.4.2012)

[10] John Ritchie, Firefox Cache Format and Extraction, http://articles.forensicfocus.com/2012/03/09/firefox-cache-format-and-extraction/ (20.4.2012)

[11] FfFormat, https://code.google.com/p/firefox-cache-forensics/wiki/FfFormat (20.4.2012)

[12] Passcape Software, Recovering Internet Explorer passwords: theory and practice http://www.passcape.com/internet_explorer_passwords (20.4.2012)