TAKING ONE BYTE AT A TIME

David Koepi

Firefox Forensics

November 27, 2010 by davidkoepi

This article is based on my research on Firefox and hands-on on an extensively run Firefox which is running on my PC. The hands-on was done on Firefox 3.6.12 running on a Windows XP SP3 machine.

~~~~~ ~~~~~

Credits and References

I like to put my credits at the start of the article because many others wrote articles on Firefox before me and I used them to help me to unravel the "mysteries" of the browser.

- MozillaZine Profile Folder – Firefox (http://kb.mozillazine.org/Profile_folder_-_Firefox) provides a detail descriptions of the files you can find in the Firefox profiles. This include previous versions of Firefox.
- FirefoxForensics – PR Time (http://www.firefoxforensics.com/research/prtime.shtml) : explains how timestamps are stored in Firefox.
- FirefoxForensics – places.sqlite::moz_historyvisits (http://www.firefoxforensics.com/research/moz_historyvisits.shtml)
- FirefoxForensics – places.sqlite::moz_places (http://www.firefoxforensics.com/research/moz_places.shtml)
- Mozilla Firefox 3 History File Format (http://www.forensicswiki.org/wiki/Mozilla_Firefox_3_History_File_Format#moz_historyvisits)
- SQLite Date Time Function (http://www.sqlite.org/lang_datefunc.html)
- It seems like whenever you google for "Firefox Forensics", you definitely sure to stumble upon Web Browser Forensics, Part 2 (http://www.symantec.com/connect/articles/web-browser-forensics-part-2) by Keith Jones and Rohyt Belani. This is a definitely must-read if you are doing forensics on web browser artifacts.

~~~~~ ~~~~~

Default Locations

WinXP:

- C:\Documents and Settings\[user]\Application Data\Mozilla\Firefox\Profiles\XXXXXXXX.default\
- C:\Documents and Settingd\[user]\Local Settings\Application Data\Firefox\Profiles\XXXXXXXX.default\Cache\

WinVista and Win7:

- C:\Users\[user]\AppData\Roaming\Mozilla\Firefox\Profiles\XXXXXXXX.default\
- C:\Users\[user]\AppData]Local\Mozilla\Firefox\Profiles\XXXXXXXX.default\Cache\

Linux:

- ~/.mozilla/firefox/XXXXXXXX.default/

Mac OS X:

- ~/Library/Application Support/Firefox/Profiles/XXXXXXXX.default/
- ~/Library/Application Support/Mozilla/Extensions
- ~/Library/Caches/Firefox/Profiles/XXXXXXXX.default/Cache/

~~~~~ ~~~~~

Cookies (cookies.sqlite)

Cookies information are located in a database table format named “moz_cookies” in cookies.sqlite3 files. Data that are of forensic interests are the host of the cookie files and the associated lastAccessed timestamp which may give indication of the time where the website was last accessed.

moz_cookies
id
name
value
host
Expiry (timestamp in Unix time)
lastAccessed (timestamp in PRTime)
isSecure
isHTTPOnly
~~~~~

(https://davidkoepi.files.wordpress.com/2010/11/screen-shot-2010-11-27-at-am-10-29-44.png)

~~~~~ ~~~~~

Downloads (downloads.sqlite)

Downloads.sqlite records details of files downloaded using Firefox. The "moz_downloads" table contains the following objects:

moz_downloads
id
name (downloaded file name)
source (source URL of downloaded files)
target (destination of downloaded files)
tempPath (temp path and filename while downloading is in progress)
startTime (PRTime)
endTime (PRTime)
state (status of downloaded files)
referrer
entityID
currBytes
maxBytes
mimeTypes
preferredApplication
PreferredAction
autoResume
downbarShow

(https://davidkoepi.files.wordpress.com/2010/11/screen-shot-2010-11-27-at-am-10-39-25.png)

I think one of the use for this table is to find out if the suspect had successfully download the files. This can be determine by investigating the 'state' field. In my analysis, a "1" in the state object indicates download is successful, "3" indicates download is cancelled, and "4" indicates download is paused. An alternate way to determine if a file is successfully downloaded is to match data in 'currBytes' with 'maxBytes' . Needless to say if the data in"currBytes" equal to "maxBytes" mean the download is completed. Data in this table will be removed when the user clears the download list.

~~~~~ ~~~~~

Form Value (formhistory.sqlite)

Form data entered in Firefox are stored in formhistory.sqlite. The “timeUsed” which helps to determine the number of times that the value was used may be helpful to investigation as it may raises the question if the value was frequently used.

moz_formhistory
id
fieldname
value
timeUsed (count of the value used)
firstUsed (first time value was used in PRTime)
lastUsed (last time value was used in PRTime)

(https://davidkoepi.files.wordpress.com/2010/11/screen-shot-2010-11-27-at-am-10-53-49.png)This table can be a wealth of information of the users’ search terms, email addresses, name and information for website registrations. One interest thing discovered in my hands-on is that search terms used in the search bar on the top right corner of Firefox are recorded in ‘fieldname’ in the ‘moz_formhistory’. with the entry as “searchbar-history”.

~~~~~ ~~~~~

Bookmarks and Internet History (places.sqlite)

Places.sqlite is probably the most important file in Firefox Forensics. In Firefox 3, Bookmarks and Internet histories are recorded in places.sqlite. In my analysis on Firefox 3.6.12, there are 10 relational tables in the database:

- moz_anno_attributes
- moz_annos
- moz_bookmarks
- moz_bookmarks_roots
- moz_favicons
- moz_historyvisits
- moz_inputhistory
- moz_items_annos
- moz_keywords
- moz_places
~~~~~

Firefoxforensics (http://www.firefoxforensics.com) did a wonderful job in mapping out the schema of the database, you can view the diagram here (http://www.firefoxforensics.com/research/firefox_places_schema.shtml). My analysis will focus on extracting meaningful data from this database on the browsing histories.

—– moz_places —–

moz_places
id (one to many relationship to "moz_historyvisits.places_id")
url
title
rev_host (hostname of the URL)
visit_count (number of visits)
hidden (a value of '1' indicates the user did not specifically visit, such as a RSS or iframe)
typed (a value of '1' indicates the user type in the URL, it can be the full URL or allows auto-complete to complete a URL)
favicon_id (one to many relationship with "moz_favicons.id")
frecency (a value calculated based on frequency and recents of the URL visits. A high value will suggest that the URL is visited more often than the other. A value of '0' will be excluded from the calculation. A few suggestions in my hands-on suggest it may be URL for downloading files and RSS. a value of '-1' suggests that it has not been calculated).
last_visit_date (timestamp of the URL last visited)

(https://davidkoepi.files.wordpress.com/2010/11/screen-shot-2010-11-27-at-am-11-09-50.png)

From this table, we can build a simple timeline based on last visited date/time (sort by the lastest visits first)

- *select moz_places.url, datetime((moz_places.last_visit_date/1000000), 'unixepoch', 'localtime') from moz_places order by moz_places.last_visit_date desc;*

However, the timeline is sorted based on the last visited time. A user may repeatedly visit the URL over the period of the Firefox history, one of the way to obtain a complete browsing history will be using data from "moz_places" and "moz_historyvisits" tables:

- *select moz_places.url, datetime((moz_historyvisits.visit_date/1000000), 'unixepoch', 'localtime') from moz_places, moz_historyvisits where moz_historyvisits.place_id = moz_places.id order by moz_historyvisits.visit_date desc;*

Obtain Google search term (sort by latest query first)

- *select moz_places.url, datetime((moz_historyvisits.visit_date/1000000), 'unixepoch', 'localtime') from moz_places,moz_historyvisits where moz_places.id = moz_historyvisits.place_id and moz_places.url*

like '%google.com%/search?q=%' order by moz_historyvisits.visit_date desc;

We can also use the following query to query if the users had typed in the URL

- *select moz_places.url, datetime((moz_historyvisits.visit_date/1000000), 'unixepoch', 'localtime') from moz_places,moz_historyvisits where moz_places.id = moz_historyvisits.place_id and moz_historyvisits.visit_type = 2 order by moz_historyvisits.visit_date desc;*

—– moz_inputhistory —–

I read an or two articles about "moz_inputhistory" explained that the table contains data relating to input URLs. However, my analysis showed this may not be true (well, I may be wrong!). My analysis suggest that it is a list of user's input text that allow autocomplete to complete the URL in the browser.

moz_inputhistory
places_id (one to many relationship with "moz_places.id")
input (user's input text)
use count (number of time the user use the same input text to autocomplete the URL)

(https://davidkoepi.files.wordpress.com/2010/11/screen-shot-2010-11-27-at-am-11-27-38.png)

One useful way is to determine the word that the user has used the text to complete the URL. For example, the user may not remember the full URL, but he said he had entered a specific text in the browser, we can use the following SQLite query to verify his claims:

- *select moz_inputhistory.input, moz_inputhistory.use_count, moz_places.url from moz_inputhistory,moz_places where moz_inputhistory.place_id = moz_places.id;*

david	2	http://davidkoepi.wordpress.com/
	2	http://www.forensicfocus.com/
thesun	2	http://www.thesun.co.uk/sol/homepage/sport/football/
www.	2	http://www.soccernet.com/

(https://davidkoepi.files.wordpress.com/2010/11/firefox-001.jpg)

—– moz_historyvisits —–

The "moz_historyvisits" detailed records of URL browsing histories and can be used with "moz_places" to create a complete timeline. The details of the table are:

moz_historyvisits
from_visit
places_id (one to many relationship to "moz_historyvisits.places_id"
visit_date (in PRTime)
visit_type: determine how user arrived at the URL '1' – user followed a link '2' – user type in the link or allow autocomplete to complete the URL '3' – user click on a bookmark '4' – embedded URL '5' – permanent redirect '6' – temporary redirect
session

(https://davidkoepi.files.wordpress.com/2010/11/screen-shot-2010-11-27-at-am-11-33-38.png)

From this table, we can determine how the user arrived at the URL. For instance, the user claimed that he was unaware of how the explicit images arrived at his computer which he claimed he had never visited. We can use the following SQLite query to determine his claim:

- *select moz_places.url, datetime((moz_historyvisits.visit_date/1000000), 'unixepoch', 'localtime'), moz_historyvisits.visit_type from moz_places, moz_historyvisits where moz_historyvisits.place_id = moz_places.id order by moz_historyvisits.visit_date desc;*

~~~~~ ~~~~~

Bookmarks (moz_bookmarks)

Bookmarks are records in "moz_bookmarks" table:
~~~~~

moz_bookmarks
id
type
fk
position: position of the bookmarks in the bookmark folder ('0' will be display at the top of the list)
parent: parent folder of the bookmarks
title
keyword_id
folder_type
dateAdded (in PRTime)
lastModified (in PRTime)

(https://davidkoepi.files.wordpress.com/2010/11/screen-shot-2010-11-27-at-am-11-41-23.png)

~~~~~ ~~~~~

Firefox Cache

In these folders, you can find the Cache Map ("_CACHE_MAP_") and 3 Cache Block files ("_CACHE_001", "_CACHE_002" and "_CACHE_003)". These 4 files are the essential files to analysis Firefox's cache files.

The Cache Map is the main file needed to reconstruct Firefox cache files. If you had read Web Browser Forensics, Part 2, (http://www.symantec.com/connect/articles/web-browser-forensics-part-2) you probably know that within the Cache Map, you probably find Cache Map buckets which contain mapping to the Cache Map records. Within the Cache Map, each Cache Record contain 4 32-bit values

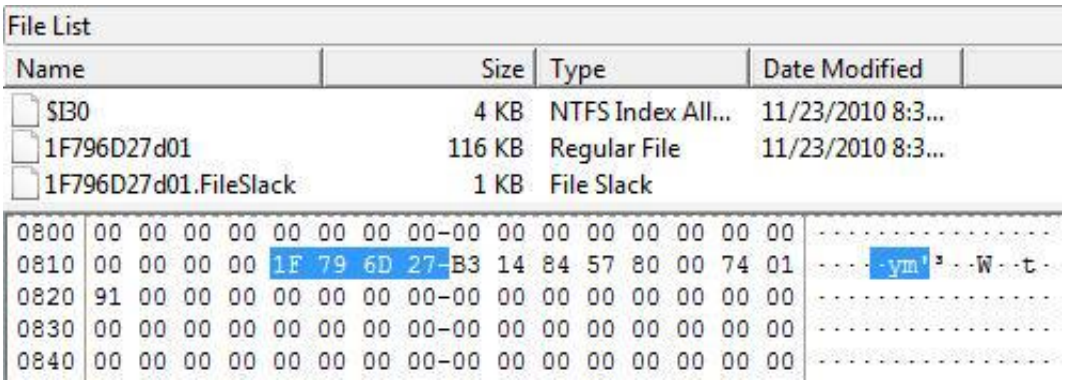
- Hash Number
- Eviction Rank
- Data Location
- Metadata Location

The 32-bit Metadata Location is bitwise-AND with 0x30000000 to obtain the metadata stored in the Cache Map or any of the 3 Cache Block file. If the resulted value from the bit-wise AND operation return a '0', the metadata are stored in the Cache Map, a value of '1' to '3' are stored in the respectively Cache Block file.

In my hands-on, the single Cache file is named "1F796D27d01". I did a search with "1F796D27" on the Cache Map and found the the offset 0x0804. The value of the Cache record is as follows:

- Hash Number = 1F796D27 (1st eight character of the cache file)
~~~~~

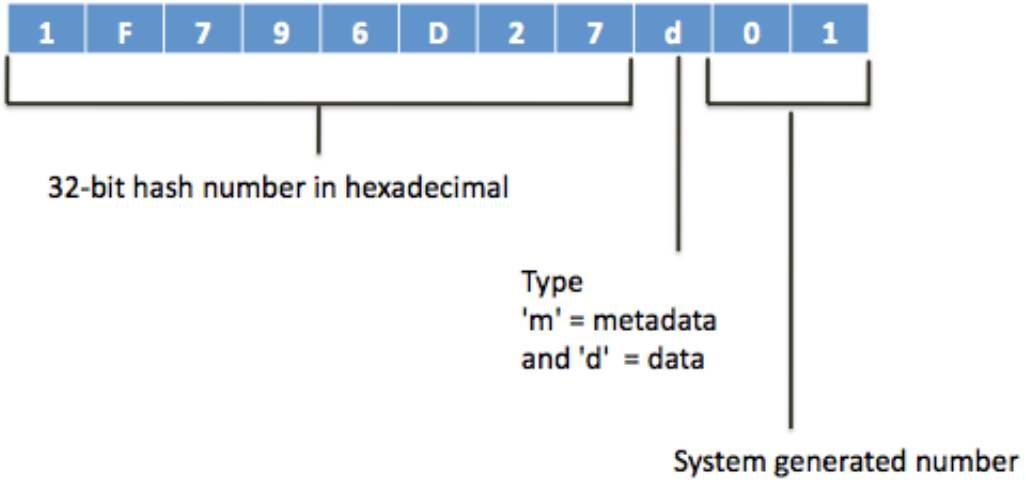
- Eviction Rank =B3148457
- Data Location =80007401
- Metadata Location = 91000000
- Cache Block locatioon = 91000000 AND 30000000 = 1 (location of metadata is stored in Cache Block 1)



(https://davidkoepi.files.wordpress.com/2010/11/firefox-02-002.jpg)

One quick and easy way to view the cache files is to use FTK Imager Lite to browse the cache files. One advantage of FTK Imager allows you to view the contents of the cache files and view the file's metadata in the "Properties" pane.

Cache files in the Cache folder are created when the content is too large to be stored in the Cache Block.In the Firefox Cache folder, Cache files are named in the following order



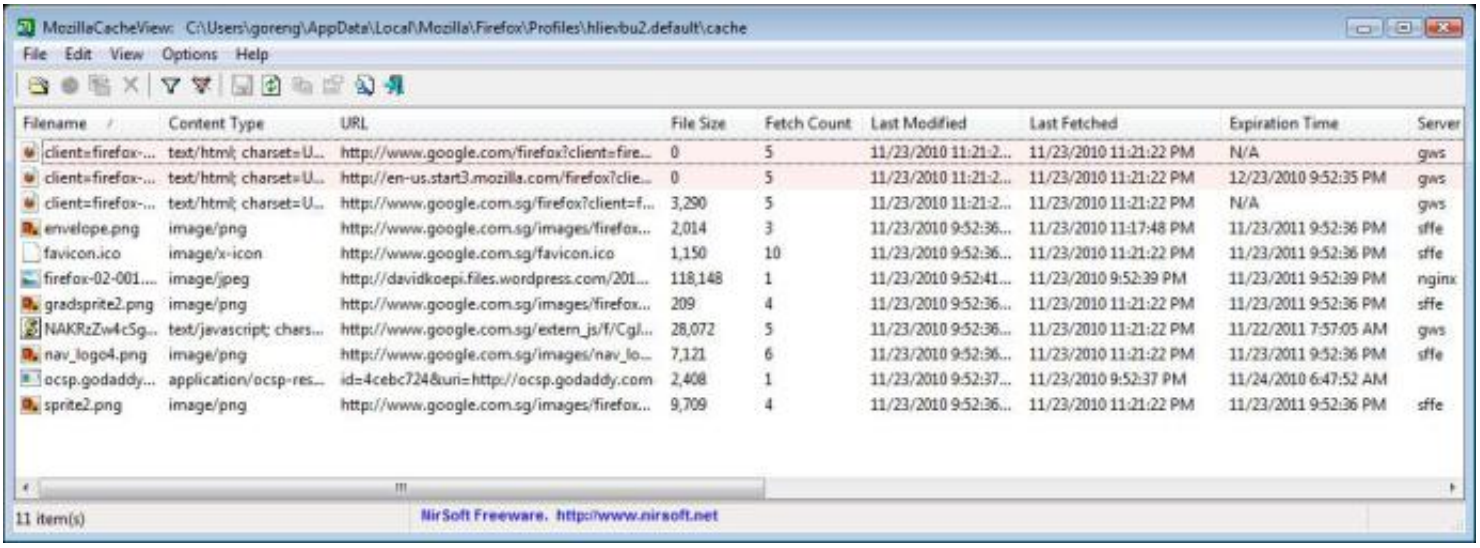
(https://davidkoepi.files.wordpress.com/2010/11/screen-shot-2010-11-27-at-pm-12-57-02.png)

In my hands-on, the filename is "1F796D27d01", it indicates that "1F796D27" is the 32-bit hash number, d represent it is a data file.

MozillaCacheView (http://www.nirsoft.net/utils/mozilla_cache_viewer.html) is a tool from Nirsoft that read and display Firefox's cache data. The metadata of forensic interests that can be retrieved are:

- Filename and filetype
- URL
- File size

- Fetch count
- Last modified / last fetch time
- Expiration time



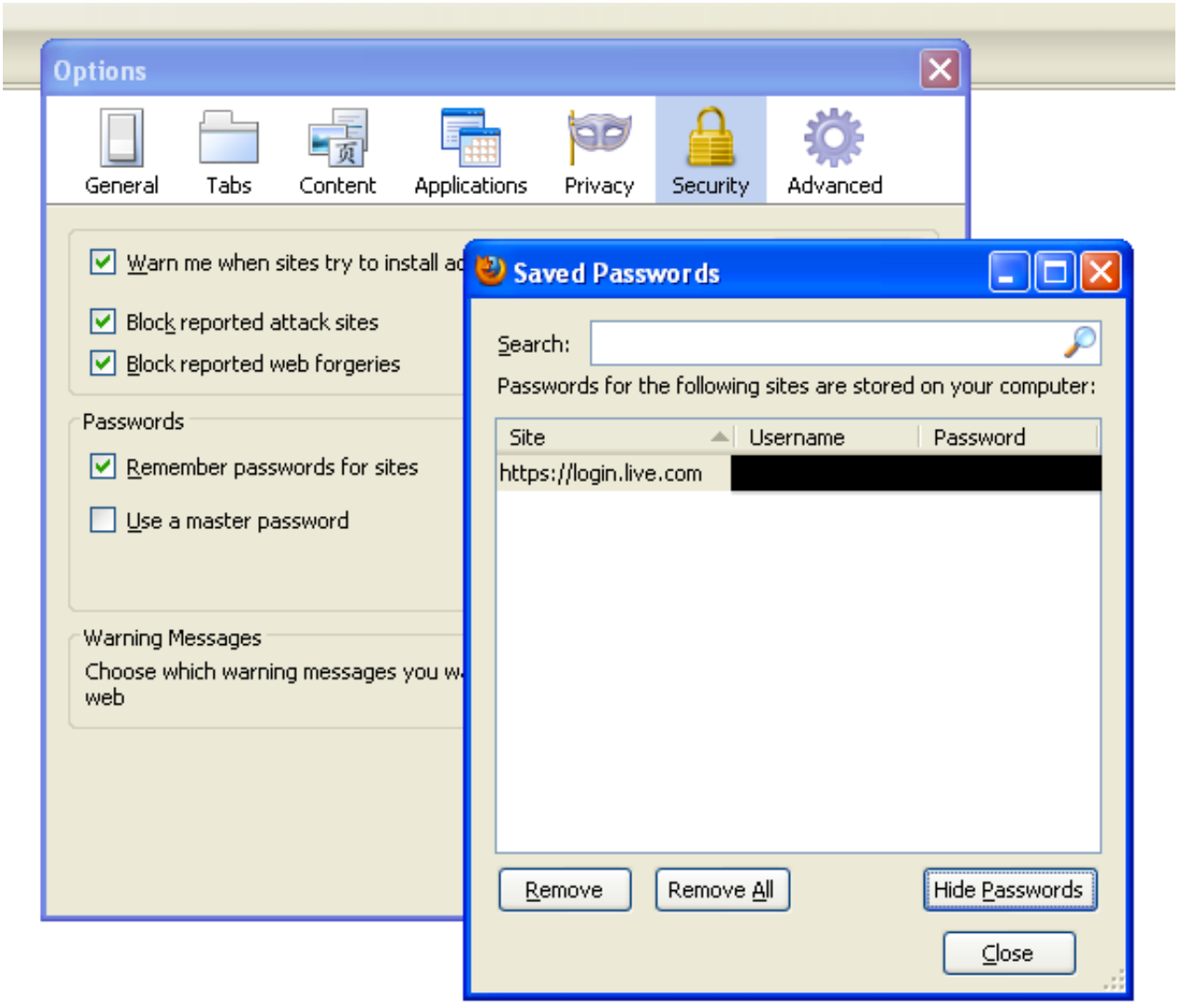
(https://davidkoepi.files.wordpress.com/2010/11/firefox-02-003.jpg)

~~~~~ ~~~~~

Saved Passwords

During my reading, I found an article on carnal0wnage.attackresearch.com (http://carnal0wnage.attackresearch.com/node/425) that explain how you can retrieve save passwords in Firefox. You can retrieve saved password saved in Firefox, if no Master Password is set. This can be done by

1. Exporting the “key3.db” and “signons.sqlite” from the user’s profiles;
2. Setup Firefox on the examiner’s machine.
3. Replace “key3.db” and “signons.sqlite” with the user’s.
4. Open up Firefox, go to Options > Security > Saved Passwords…
~~~~~



(https://davidkoepi.files.wordpress.com/2010/11/screen-shot-2010-11-27-at-pm-10-47-10.png)

~~~~~ ~~~~~

This Post was posted in Browser Forensics, Forensics on Windows. Bookmark the permalink.
~~~~~