Analyzing Safari 2.x Web Browser Artifacts using SFT.

Table of Contents

Introduction

Many computer forensic investigations include gathering evidence about a user's web browser configuration and usage. This paper is intended to help investigators understand the structure and contents of the files created and used by the Safari web browser (v2.4.0) on MacOSX 10.4, and illustrate how data can be extracted from these files using Safari Forensic Tools.

Safari is a feature rich web browser based on the WebKit framework and has come pre-installed on Mac systems since OSX 10.2.8. The different OSX revisions ship with different versions of Safari. The version can be incremented via patching (OSX Software Update), or by downloading and installing the latest build. The Safari and WebKit Version Information matrix provided by Apple shows the available Safari revisions. [1]

Safari is similar to most commonly available web browsers. It stores a history of visited URLs, cookies received from websites, files downloaded, and caches viewed content to local disk to speed up subsequent page loads. Users can bookmark URLS and have the option of storing website login credentials for future use.

Safari Forensic Tools

The Safari Forensic Tools (SFT) is a collection of open source programs that are designed to parse Safari related files and output the contents in a text form that can be easily included in a forensic report. The SFT tools are written in Objective-C. They will compile under Linux by using the GNUStep framework, and under OSX using the Foundation framework . SFT can be downloaded from http://jafat.sourceforge.net

OSX Property List files

The OSX operating system and many OSX applications store persistent data in property list files, which are designated by the .plist file extension. Applications can use both XML (ASCII text) format and binary format property list files.
The application data in the property list file is stored as serialized Foundation Kit objects such as: NSString, NSDictionary, NSNumber, NSDate, NSData and NSArray.

1: http://developer.apple.com/internet/safari/uamatrix.html

Safari Related Files

By default the Safari package is installed in the /Applications/Safari.app/Contents directory at the root of the filesystem. The package consists of the Safari program binary under the MacOS subdirectory, supporting files (button images etc.) under the Resources subdirectory, a version.plist file and a Info.plist file.

The two files in this location that may be of interest to forensic examiners are the version.plist and Info.plist files. The version.plist file is a XML based plist file which contains detailed information about the installed Safari version.

The Info.plist file is an XML based plist file which contains system wide configuration information. The biggest portion of the file is typically the CFBundleDocumentTypes key in the top level NSDictionary. This key contains nested NSArrays and NSDictionaries with information about the file types supported by the browser.

OSX is a multi-user operating system, therefore Safari stores the per-user preferences and data files under each user's home directory,which are usually contained under the **/Users/** top level directory. Table 1 shows the location of the per-user Safari plist data files, and other related directories of interest.

Table 1: Per-User Safari data files

File/Location (~ is user homedir)	Description
~/Library/Preferences/com.apple.Safari.plist	Binary plist containing General Safari preferences
~/Library/Preferences/com.apple.internetconfigpriv.plist	Binary plist containing Internet preferences
~/Library/Preferences/com.apple.internetconfig.plist	Binary plist containing Internet preferences
~/Library/Safari/Downloads.plist	XML plist containing a history of downloaded files
~/Library/Safari/History.plist	Binary plist containing a history of sites visited
~/Library/Safari/Bookmarks.plist	Binary plist of Bookmarked sites
~/Library/Cookies/Cookies.plist	Binary plist with Cookie data from sites visited
~/Library/Safari/Icons	Directory of cached “favicons” from sites visited
~/Library/Safari/Form Values	Encrypted file containing data entered into web forms.
~/Library/Caches/Safari/	Directory of cached data from sites visited.

Safari Preferences Files

The Safari general preferences (or defaults) are stored in the file ~/Library/Preferences/com.apple.Safari.plist.
This preferences file contains information about options selected in the Safari-> Preferences window menu, information about if/how it should be displayed in the toolbar, and window sizes.
It is a binary plist file that contains a NSDictionary of configuration key/value pairs. Some of the values can be additional NSDictionary and NSArray objects.

My testing revealed that in addition to the default values, key/value pairs are added to this file only if specific non-default options are selected.

Some key/value pairs found in this preferences file that are significant for forensic investigators are:

Table 2: Safari Preference File keys

NSDictionary key	Significance
LastRunVersion	The value of this key lists the version of Safari.
WebKitPrivateBrowsingEnabled	When the value of this key is set to "yes" Private Browsing is enabled (Default is “No”)
CachedBookmarksFileDateSeconds	Second since epoch of the modified time of the Bookmarks.plist file
CachedBookmarksFileSize	Size of the Bookmarks.plist file
WebIconDatabaseDirectoryDefaultsKey	The location where Safari caches the “favicons” of visited sites.

Additional configuration information of interest to forensic investigators can be found intermixed with other system preferences in the com.apple.internetconfigpriv.plist and com.apple.internetconfig.plist references files in the ~Library/Preferences directory.:

Table 3: Other preference files

NSDictionary key	Significance	File Location
WWWHomePage	The startup Homepage	~/Library/Preferences/com.apple.internetconfigpriv.plist
DownloadFolder	Location to save downloaded files (~/Desktop by default)	~/Library/Preferences/com.apple.internetconfig.plist

The contents of these Safari preferences files can be viewed by using the ***pref_parser*** program included with the Safari Forensic Tools. The example in Figure 1 shows the output from running the ***pref_parser*** utility on the com.apple.Safari.plist file. Note that several preferences included in the file have been omitted to keep the output terse in this example.

Figure 1.

```
jake$: pref_parser com.apple.Safari.plist

CachedBookmarksFileSize:        22350
WebIconDatabaseDirectoryDefaultsKey:    ~/Library/Safari/Icons
BuiltInBookmarksDate:   2006-03-12 19:51:51 -0500
WebKitPrivateBrowsingEnabled:   NO
CachedBookmarksFileDateSeconds: 1172498512
LastVersionRun: 419.3
```

Browser History

Safari stores the user's browsing history in the file ~/Library/Safari/History.plist
This binary plist contains a NSDictionary with a key of “WebHistoryDates” and the value of the key is a NSArray of NSDictionaries containing the history data.

Table 4 shows the structure of the inner NSDictionaries that contain the history data.

Table 4: History.plist inner keys

Data Type	Key	Significance
NSString	(null)	Visited URL
NSString	title	Title of the page
NSDate	lastVisitedDate	Last date/time the URL was visited
NSData	visitCount	Number of times the URL was visited

The ***safari_hist*** program in the SFT parses the History.plist file and presents the contents in TAB delimited form as shown in Figure 2.

Figure 2:

```
jake$: ./safari_hist History.plist
URL     Last Visit Date/Time    Number of visits        Page Title
http://www.google.com/  2006-12-03 15:23:08 -0500       10      Google
http://livepage.apple.com/      2006-12-03 15:23:03 -0500       7       Apple - Start
http://developer.apple.com/technotes/tn/tn2003.html     2006-12-03 14:05:55 -0500    1  Technical Note TN2003: Moving Your Code to Mac OS X
http://developer.apple.com/technotes/tn2002/tn2071.html 2006-12-03 14:03:25 -0500    1  Technical Note TN2071: Porting Command Line Unix Tools to Mac OS X
http://www.mactel-linux.org/    2006-12-03 11:14:29 -0500       1       Main Page - Mactel-Linux
http://www.mac-intel.org/       2006-12-03 11:14:03 -0500       1       Welcome to mac-intel.org
http://www.mac-tel.org/ 2006-12-03 11:13:58 -0500       1       Welcome to mac-tel.org
```

Downloads History

The XML plist file ~/Library/Safari/Downloads.plist contains records of the downloaded files. It consists of a NSDictionary with a key named “DownloadHistory” that contains an NSArray of NSDictionaries. The inner NSDictionaries contain key/value pairs with data about the downloaded files as shown in Table 5.

Table 5: Downloads.plist inner keys

NSDictionary Key	**Significance**
DownloadEntryProgressBytesSoFar	Number of bytes Downloaded when the record was written.
DownloadEntryPostPath	-
DownloadEntryIdentifier	A UUID (Universally Unique Identifier) for the download.
DownloadEntryPath	The location of the downloaded file.
DownloadEntryProgressTotalToLoad	The total size of the original file (in bytes).
DownloadEntryURL	The URL of the original file.

The values of “DownloadEntryProgressBytesSoFar” and “DownloadEntryProgressTotalToLoad” will be equal in entries for successful downloads.

My testing showed that in cases where the download was interrupted or canceled by the user, the additional key/value pair: DownloadEntryErrorCodeDictionaryKey: -999 was present, and the inner NSDictionaries had values that were NSDictionaries containing additional information about the status of the failed download.

The SFT package contains the utility ***safari_download*** to parse the Downloads.plist file and print the contents of each download record as shown in Figure 3.

The “Status:” portion of the safari_download tool output is derived data. The tool checks for the existence of the “DownloadEntryErrorCodeDictionaryKey: -999” key/value pair. If it is present, it then compares the values of the “DownloadEntryProgressBytesSoFar” and “DownloadEntryProgressTotalToLoad” keys.

If these two values are equal, it shows the user chose to cancel the download. If the value of “DownloadEntryProgressBytesSoFar” is less than the value of “DownloadEntryProgressTotalToLoad”, this indicates that the transfer was somehow interrupted and did not complete.

Figure 3:

```
jake$: ./safari_download  Downloads.plist

DownloadEntryProgressBytesSoFar: 18411854       DownloadEntryPostPath: /Volumes/Firefox/Firefox.app
DownloadEntryIdentifier: 27D934DC-F026-4002-A31D-58707BB93C06    DownloadEntryPath:
~/Desktop/Firefox 2.0.dmg    DownloadEntryURL: http://ftp-
mozilla.netscape.com/pub/mozilla.org/firefox/releases/2.0/mac/en-US/Firefox%202.0.dmg
DownloadEntryProgressTotalToLoad: 18411854       Status: Completed


DownloadEntryProgressBytesSoFar: 166530 DownloadEntryPostPath: ~/Desktop/dcfldd-1.3.4.tar
DownloadEntryIdentifier: 17FA3845-851C-4FDD-8469-ED2D2FCEEFFE   DownloadEntryPath:
~/Desktop/dcfldd-1.3.4.tar.gz.download/dcfldd-1.3.4.tar.gz    DownloadEntryURL:
http://jaist.dl.sourceforge.net/sourceforge/dcfldd/dcfldd-1.3.4.tar.gz  DownloadEntryProgressTotalToLoad:
166530        Status: Completed


DownloadEntryProgressBytesSoFar: 398872 DownloadEntryPostPath: ~/Desktop/netcat-0.7.1.tar
DownloadEntryIdentifier: 293F512C-1E5F-468C-B420-77A2EE2FDDB7   DownloadEntryPath:
~/Desktop/netcat-0.7.1.tar.gz.download/netcat-0.7.1.tar.gz    DownloadEntryURL: http://superb-
west.dl.sourceforge.net/sourceforge/netcat/netcat-0.7.1.tar.gz    DownloadEntryProgressTotalToLoad:
398872        Status: Completed
```

Bookmarks File

Safari stores URLs that the user bookmarks in the ~/Library/Safari/Bookmarks.plist file. This binary plist file contains a top-level NSDictionary that contains several levels of nested NSArrays and NSDictionaries under the “Children” key.

The ***safari_bm*** tool in SFT will parse the Bookmarks.plist file and print out the contents as shown in Figure 4. The nested levels in the output reflect the structure and organization of the bookmarks. For example a user can have bookmarks, as well as labeled bookmark folders containing groupings of bookmarks.

Figure 4:

```
jake$: ./safari_bm Bookmarks.plist
Title: BookmarksBar
        Apple
                http://www.apple.com/
        .Mac
                http://www.mac.com/
        Apple Hot News
                feed://www.apple.com/main/rss/hotnews/hotnews.rss
        Wired
                feed://wired.com/news/feeds/rss2/0,2610,3,00.xml
Title: BookmarksMenu
Title: Address Book
Title: Bonjour
Title: History
Title: All RSS Feeds
```

Cookies File

Cookie information provided by websites is stored in ~/Library/Cookies/Cookies.plist . This XML plist file contains an NSArray of NSDictionaries with key/value pairs for each cookie entry. Each cookie entry contains the fields shown in Table 7.

Table 7: Cookies.plist keys/values

DataType	**Key**	**Significance**
NSDate	Created	Date the cookie entry was created
NSString	Domain	URL of the site
NSString	Expires	Date cookie expires
NSString	Name	Title of the site
NSString	Path	URL Path
NSString	Value	Cookie data

The ***safari_cookies*** utility parses the Cookies.plist file and prints out the contents of each cookie record. The NSDate *Created* key is converted to a readable time format, and output with the original value printed in parenthesis.

Figure 5:

```
jake$: ./safari_cookies Cookies.plist
Path:   /
Domain: .mozilla.com
Name:   __utma
Created:        2006-11-02 12:33:25 -0500 (184181605.669188)
Expires:        2038-01-18T00:00:00Z
Value:  183859642.1905229267.1162488806.1162488806.1162488806.1

Path:   /
Domain: .apple.com
Name:   s_vi
Created:        2006-08-08 14:04:28 -0400 (176753068.882685)
Expires:        2011-08-07T18:04:35Z
Value:  [CS]v1|44D8D23300003601-A000C57000003E7[CE]
```

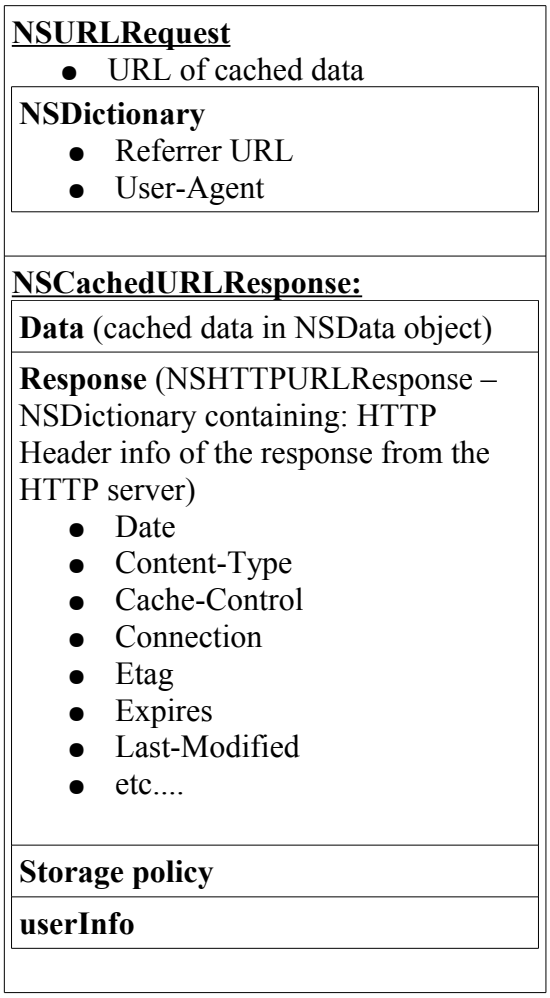
Browser Cache

Safari stores cached website content such as HTML, javascript, images, etc. under the ~/Library/Caches/Safari/ directory. This directory contains two levels deep of subdirectories named with 2 digit numbers. (ex.00, 01, 02 ...) The files in these directories contain the cached data and are named with a 10 digit number, a dash, and another 10 digit number and the .cache extension. (ex: 3845526787-0994524173.cache).

These cache files are OSX Archive files that store large amounts of data in a platform independent stream of bytes. The first portion of the cache file is a NSURLRequest object containing the URL of the source of the cached data, and an NSDictionary with the the referrer URL, and the User-Agent reported by the Safari browser at the time of the request.

The next portion of the cache file is a NSCachedURLResponse object. This object is comprised of a NSURLResponse object containing among other things, the URL of the cached request, an NSData object containing the cached data (html,js,JPG,GIF, etc), and an NSDictionary with HTTP header fields from the web server. This NSDictionary could contain any RFC 2616 HTTP header entry.
Figure 6 is a visual representation of the cache file contents.

Figure 6:



The ***safari_cache*** utility included with SFT will parse the browser .cache files and output the URL, the referring URL, and the Date HTTP Header information from the server in TAB delimited format as shown in Figure 7. The utility also extracts the cached data to a file named "cachedata.out" in the local directory. In this example the file cachedata.out is a JPG file that can be loaded into a JPG viewer.

Figure 7:

```
$ ./safari_cache 1294766672-0794204939.cache
URL     Referrer          Cached date/time
http://a1.phobos.apple.com/r30/Features/f6/6d/a5/dj.vmpykbvc.jpg http://www.apple.com/startpage Tue, 27 Feb 2007 23:15:22 GMT
```

The ***safari_cache*** utility will only run on OSX systems. OSX uses a proprietary NSArchiver format that is unreadable by GNUstep.

Icon Cache

If a visited site has a "favicon.ico" defined, Safari will cache the Icon in the ~/Library/Safari/Icons directory (by default). This directory contains a subdirectory structure similar to that of the browser cache, with two levels of subdirectories named with 2 digit numbers. (ex: 00, 01, 02 etc). The Icon cache files are also named similarly to the browser cache files with a 10 digit number, a dash, and another 10 digit number and the .cache extension.
(ex: 3845526787-0994524173.cache).
These icon cache files are NSArchive files which typically contain an NSString object with the URL of the site and an NSData object containing the binary data of the icon image. The icon image is stored as a TIFF file.

The ***safari_icon_osx*** utility in SFT parses the icon cache files created by Safari and outputs the URL of the cached icon and the date the icon was cached in TAB delimited format as shown in Figure 8. This utility also extracts the cached icon image to a file named "icon_output.ico" in the current directory. The icon_output.ico file can be viewed using any image viewer that supports the TIFF image format.

Figure 8:

```
$ ./safari_icon_osx 4412568132-9582434816.cache
http://www.weather.com/favicon.ico          2007-02-26 13:03:06 -05:00
```

The safari_icon_osx utility can only be run on OSX systems. OSX uses a proprietary NSArchiver format that is unreadable by GNUstep.

Private Browsing

Private Browsing is a new feature in Safari that limits the amount of information Safari saves to the hard disk.

When Private Browsing is enabled by the user, Safari present a message saying:

"*When private browsing is turned on, web pages are not added to the history, items are automatically removed from the Downloads window, information isn't saved for AutoFill (including names and passwords), and searches are not added to the pop-up menu in the Google search box. Until you close the window, you can still click the Back and Forward button to return to web pages you have opened*."

In addition to the above features of Private Browsing, data is not saved in the browser cache directory, and Cookies are removed when Private Browsing is turned off. The Cookies.plist file gets populated with Private Browsing enabled, so it may be possible for an investigator to recover the deleted Cookies.plist file from a Private Browsing session.

An investigator can sometimes determine if Private Browsing was enabled.
The ~/Library/Preferences/com.apple.Safari.plist file will contain the "WebKitPrivateBrowsingEnabled Yes" key/value pair if Private Browsing was enabled in the last session, and the browser has not been restarted because "Private browsing is always turned off when you open Safari, even if it was on when you last quit Safari" [2]

Icon data is still saved to the Icon Cache even when Private Browsing is enabled, if the user visits a site that doesn't have an Icon previously cached by Safari. This icon data can be retrieved, and the metadata analyzed to determine if a user visited a particular site when Private Browsing was enabled.

Stored Website Passwords

OSX implements a system-wide keychain as a secure location for users and applications to store login credentials, encryption keys and certificates. The keychain is encrypted with 3DES using a master password to protect its contents. Safari uses this keychain mechanism to "remember" website login usernames and passwords and auto-fill them when a user re-visits a site, assuming this non-default functionality has been enabled in the Preferences menu.
There are no tools in the SFT package to access the keychain.

AutoFill Form Values

If a user has enabled the AutoFill WebForms option in the Safari Preferences, the browser will store web form input values in the ~/Library/Safari/Form Values file, which appears to be encrypted.

Browser Plug-ins

Plug-ins can be used to enhance and/or alter the Safari browser. A list of Safari plug-ins that are installed on the system is located in the file:
/Applications/Safari.app/Contents/Resources/English.lproj/Plug-ins.html

---

2 http://docs.info.apple.com/article.html?path=Safari/2.0/en/ibr1069.html

References

Apple Inc. “File system Overview”
http://developer.apple.com/documentation/MacOSX/Conceptual/BPFileSystem/index.html

Apple Inc. “OSX Technology Overview”
http://developer.apple.com/documentation/MacOSX/Conceptual/OSX_Technology_Overview/index.html

Apple Inc. “Security in MacOSX”
http://images.apple.com/macosx/pdf/Mac_OS_X_Security_TB.pdf

Apple Inc. “Safari and WebKit Version Information'
http://developer.apple.com/internet/safari/uamatrix.html

Apple Inc. “Resource Programming Guide: Property Lists”
http://developer.apple.com/documentation/Cocoa/Conceptual/LoadingResources/MOSXResources/chapter_2_section_5.html

Apple Inc. “Safari Private Browsing”
http://docs.info.apple.com/article.html?path=Safari/2.0/en/ibr1069.html

Apple Inc. “About Property Lists”
http://developer.apple.com/documentation/Cocoa/Conceptual/PropertyLists/Articles/AboutPropertyLists.html

Apple Inc. “Archives”
http://developer.apple.com/documentation/Cocoa/Conceptual/Archiving/index.html

Apple Inc. “NSArchiver Class References”
http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSArchiver_Class/Reference/Reference.html

Apple Inc. “NSCachedURLResponse Class Reference”
http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSCachedURLResponse_Class/Reference/Reference.html

GnuStep
http://www.gnustep.org

Safari Forensic Tools
http://jafat.sourceforge.net

Webkit Open Source Project
http://webkit.org/