Hindsight User Guide

Internet History Forensics for Google Chrome

Hindsight Version: 1.3.0
Author: Ryan Benson
Date: 2014-11-10

Overview

Hindsight is a free tool for analyzing the browsing history of the Google Chrome web browser. It can collect a number of different types of Chrome artifacts, including URLs, download history, bookmarks, autofill records, HTTP cookies, and Local Storage records (HTML5 cookies). Once the data is extracted from each file, it is correlated with data from other history files and placed in a timeline.

After the data is extracted, Hindsight runs a number of plugins against the data to try to further interpret what it has found. A plugin is a separate Python file that Hindsight runs that performs a specific action, such as parsing a particular URL or cookie. Plugins could perform actions that just use local resources (such as parsing Google Analytics tracking cookies) as well as connecting to and using remote resources (such as looking up visited URLs to flag ones associated with malware or phishing). Users can choose which plugins to run and are welcome to submit ideas for new ones or to create their own.

The last piece of Hindsight is the reporting. Once the data has been collected and the plugins have run, Hindsight has a number of output options; the default option is to create an .xlsx spreadsheet. The xlsx format was chosen for a number of reasons, including the ability to do advanced filtering and the fact that most end users are already familiar with it. Whenever possible, Hindsight tries to group similar types of data from different browser artifacts into one column, enabling the reader to more easily scan the data and quickly understand it.

While a spreadsheet is likely the most end-user friendly output, Hindsight also can output data in ways that are easier for computers to parse: SQLite and JSON. The aim of these two output formats is to make the tool more flexible and easier to integrate into other programs or complex workflows.

Hindsight is an open source tool, which means anyone who is interested can view how it works and even modify it. It is written in Python, and can run on Windows, Linux, or Mac systems where Python (and the required packages) are installed.

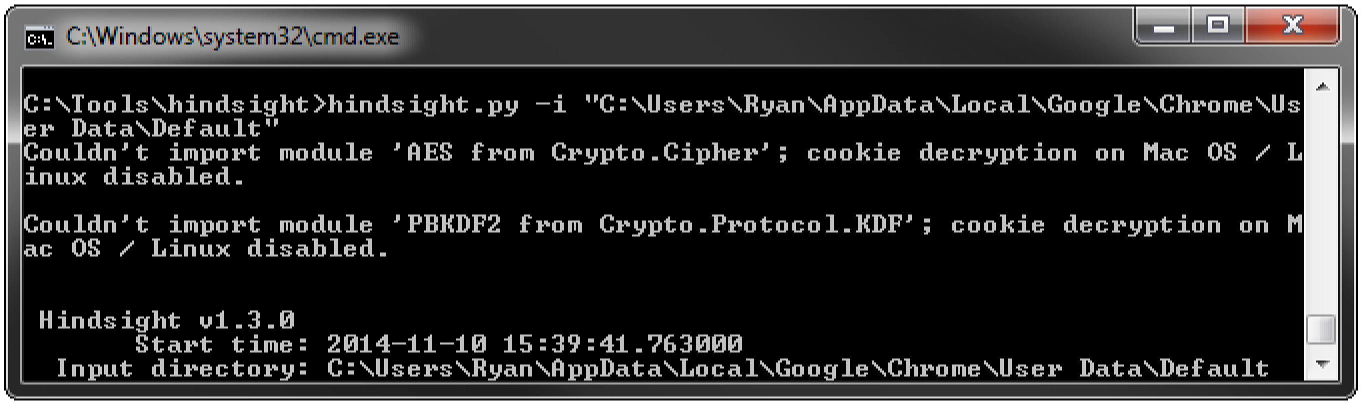
Installation and Prerequisites

Hindsight is written in Python and requires the Python 2.7 interpreter to be installed on the analysis workstation (available at python.org). Hindsight also needs a number of additional Python packages not included in the default interpreter installation, and Hindsight plugins may also require extra packages. Most packages are available via the Python Package Index (https://pypi.python.org/pypi) and can be installed at the command line via pip.

After installing pip, it can be used to install some of the required packages. In the commands below it is assumed the user is using Windows and that Python 2.7 was installed to C:\Python27. XlsxWriter is required if the user wants to create XLSX report spreadsheets, and keyring, pycrypto, and PyWin32 are all used for decoding encrypted cookies (each of the three commands below should be entered on a single line, excluding >):

```
> C:\Python27\Scripts\pip.exe install xlsxwriter
> C:\Python27\Scripts\pip.exe install keyring
> C:\Python27\Scripts\easy_install.exe
   http://www.voidspace.org.uk/downloads/pycrypto26/pycrypto-2.6.win32-
   py2.7.exe
```

Lastly, install PyWin32 using the installer from http://sourceforge.net/projects/pywin32/files/pywin32/Build%20219/pywin32-219.win32-py2.7.exe/download. If any of these packages are not installed, Hindsight will still run, just with diminished capabilities. It will warn the user as it starts if any packages/features are missing.



```
C:\Tools\hindsight>hindsight.py -i "C:\Users\Ryan\AppData\Local\Google\Chrome\Us
er Data\Default"
Couldn't import module 'AES from Crypto.Cipher'; cookie decryption on Mac OS / L
inux disabled.

Couldn't import module 'PBKDF2 from Crypto.Protocol.KDF'; cookie decryption on M
ac OS / Linux disabled.


 Hindsight v1.3.0
        Start time: 2014-11-10 15:39:41.763000
  Input directory: C:\Users\Ryan\AppData\Local\Google\Chrome\User Data\Default
```

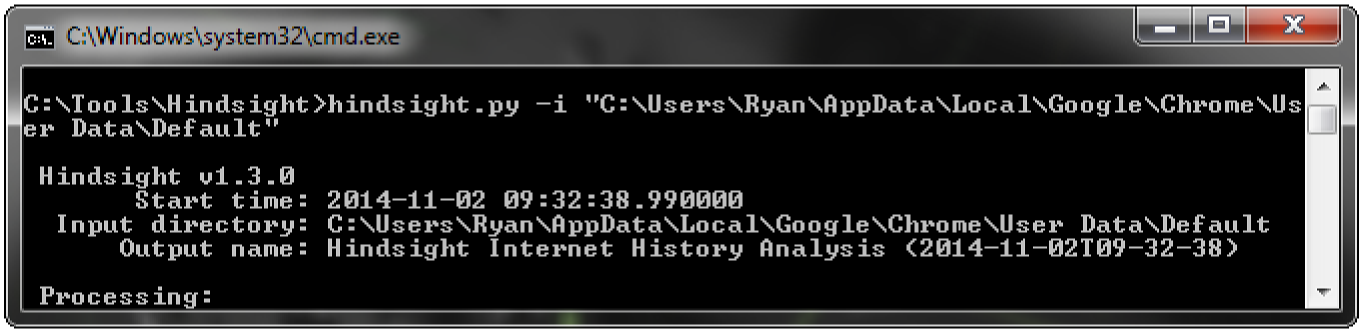
Figure 1: Running Hindsight without some packages installed

The last step is to prepare Hindsight itself, which is very simple. Download Hindsight (https://github.com/obsidianforensics/hindsight/archive/master.zip) and extract the contents of the .zip file into the directory where Hindsight will run from. After extracting the files the directory should have a `hindsight.py` file and a subdirectory called `plugins` (along with some documentation material). Hindsight is now ready to run. The last optional step is to add the directory containing Hindsight to the analysis system's PATH variable; otherwise the user will have to navigate to the folder containing Hindsight before running it.

Running Hindsight

Hindsight is a script that runs from a command line interface. In Windows, the command line is typically accessed via the Command Prompt (cmd.exe). Linux and Mac users can use the Terminal application. Command line programs can be intimidating to users unfamiliar with them, but are actually fairly easy to use. Many command line programs are run by entering the program's name into the command prompt followed by some options that tell the program what to do.

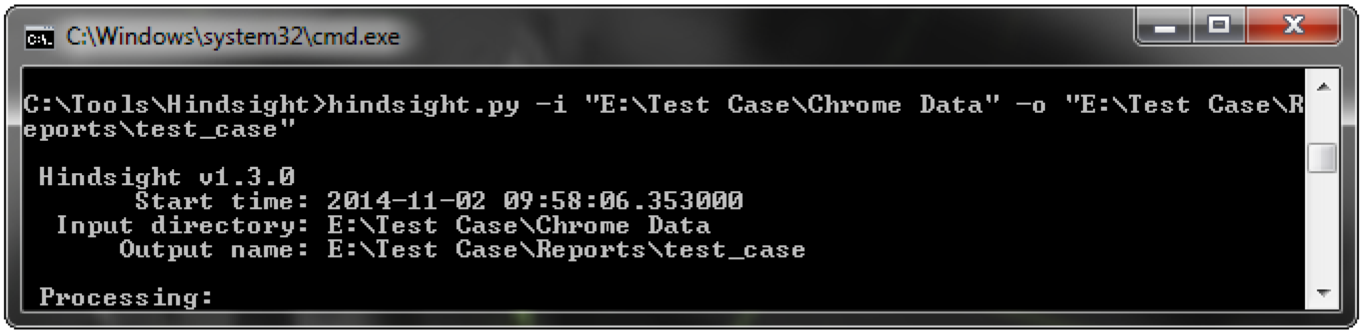
Running Hindsight from the command line is fairly straightforward, as it has few options and only one is mandatory. This option is for a directory to process. This could be the directory of the local Chrome installation on the system, or an evidence directory containing files collected from another computer. Specify the location which contains the Chrome data files using the `-i` (input) option followed by the path to the directory.



```
C:\Tools\Hindsight>hindsight.py -i "C:\Users\Ryan\AppData\Local\Google\Chrome\Us
er Data\Default"

 Hindsight v1.3.0
        Start time: 2014-11-02 09:32:38.990000
  Input directory: C:\Users\Ryan\AppData\Local\Google\Chrome\User Data\Default
      Output name: Hindsight Internet History Analysis (2014-11-02T09-32-38)

 Processing:
```

Figure 2: Running Hindsight with only –`i` option on local Chrome installation

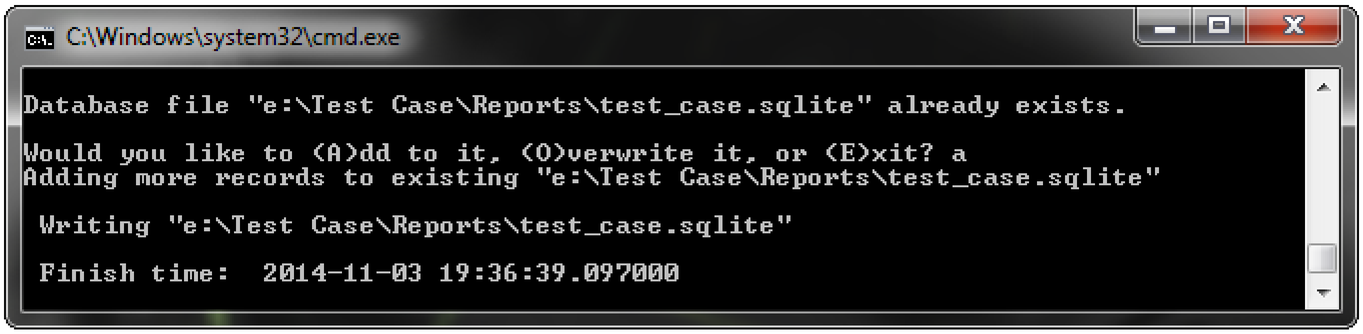
By default, Hindsight will save its output in an .xlsx spreadsheet in the directory from which it was run, with a base name of "Hindsight Internet History Analysis (yyyy-mm-ddThh-mm-ss)" using the time that Hindsight started executing. If one desires a more descriptive name or would like to save it in a different location, the `-o` (output) option can be used to specify the name (with or without a directory). If you specify a file with the `-o` option that already exists, Hindsight will overwrite that file, so please use with caution.



```
C:\Windows\system32\cmd.exe

C:\Tools\Hindsight>hindsight.py -i "E:\Test Case\Chrome Data" -o "E:\Test Case\R
eports\test_case"

 Hindsight v1.3.0
       Start time: 2014-11-02 09:58:06.353000
  Input directory: E:\Test Case\Chrome Data
      Output name: E:\Test Case\Reports\test_case

 Processing:
```

Figure 3: Running Hindsight with `-i` and `-o` options

The default output format for Hindsight is an .xlsx spreadsheet. To select a different file format, use the `-f` flag and specify one of the supported formats (SQLite, JSON, or XLSX). The SQLite format is special, in that it is the only format currently that can be appended to. If the user wants to process multiple Chrome installations and combine them, they have to use the `-o` option to specify the same output file and the `-f` option to select SQLite. If the SQLite file exists, Hindsight will prompt the user with what to do: add to it, overwrite it or exit. Additionally, the user can set the `-m` flag to answer this question at invocation; this option exists to aid in creating scripts.

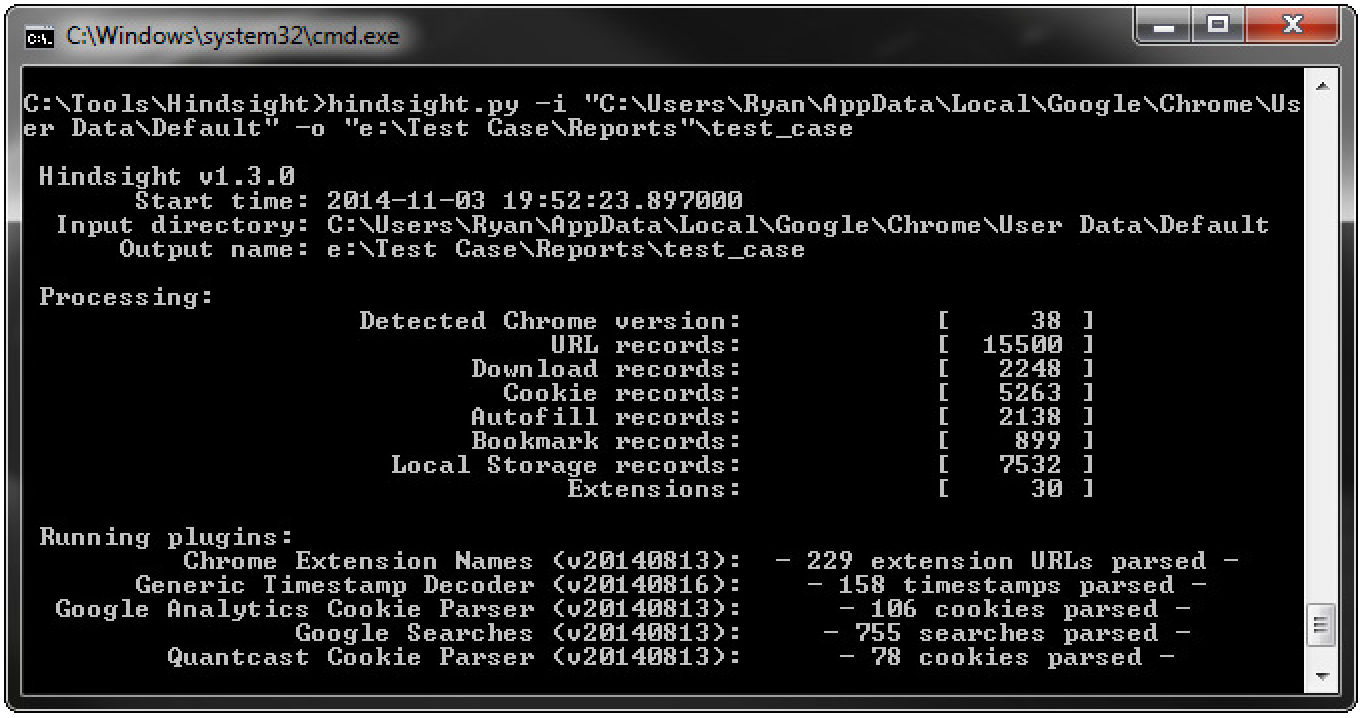


```
C:\Windows\system32\cmd.exe

Database file "e:\Test Case\Reports\test_case.sqlite" already exists.

Would you like to (A)dd to it, (O)verwrite it, or (E)xit? a
Adding more records to existing "e:\Test Case\Reports\test_case.sqlite"

 Writing "e:\Test Case\Reports\test_case.sqlite"

 Finish time:  2014-11-03 19:36:39.097000
```

Figure 4: Running Hindsight with `-o` specifying an existing database

The last three command line options for Hindsight are `-h`, `-r` and `-l`. `-h` displays a simple help message; this message is also displayed when Hindsight fails to understand the options the user entered. Adding the `-r` option allows the use of remote resources. By default, Hindsight will not connect to any remote resources (such as web sites or APIs), as in some instances investigators do not want any data from their cases going out to a third party. However, Hindsight can do a more complete analysis of the collected files if it is allowed to query other services. For example, the *Safe Browsing API Lookup* plugin checks each URL against Google's Safe Browsing service to check for known malware or phishing sites. Hindsight creates and audit log when it runs; `-l` is the option to change where this log is saved (the default log is called `hindsight.log` and is saved in the same directory as `hindsight.py`).

After the command line options are set and the program is launched, Hindsight does its best to keep the user apprised of the program's progress. The first thing Hindsight does is attempt to determine what version of Chrome artifacts are in the input directory. This is necessary because Chrome updates so frequently, and the databases that store artifacts often change in new versions of the browser. Hindsight uses these differences in database schema to figure out the likely version. Once this is done, Hindsight begins processing Chrome's numerous artifacts. An entry is displayed for each artifact type, along with how many of those artifacts Hindsight has processed.



```
C:\Tools\Hindsight>hindsight.py -i "C:\Users\Ryan\AppData\Local\Google\Chrome\Us
er Data\Default" -o "e:\Test Case\Reports"\test_case

 Hindsight v1.3.0
       Start time: 2014-11-03 19:52:23.897000
  Input directory: C:\Users\Ryan\AppData\Local\Google\Chrome\User Data\Default
      Output name: e:\Test Case\Reports\test_case

 Processing:
                    Detected Chrome version:         [    38 ]
                                URL records:         [ 15500 ]
                           Download records:         [  2248 ]
                             Cookie records:         [  5263 ]
                           Autofill records:         [  2138 ]
                           Bookmark records:         [   899 ]
                      Local Storage records:         [  7532 ]
                                 Extensions:         [    30 ]

 Running plugins:
          Chrome Extension Names (v20140813):  - 229 extension URLs parsed -
       Generic Timestamp Decoder (v20140816):   - 158 timestamps parsed -
  Google Analytics Cookie Parser (v20140813):     - 106 cookies parsed -
                 Google Searches (v20140813):    - 755 searches parsed -
         Quantcast Cookie Parser (v20140813):     - 78 cookies parsed -
```

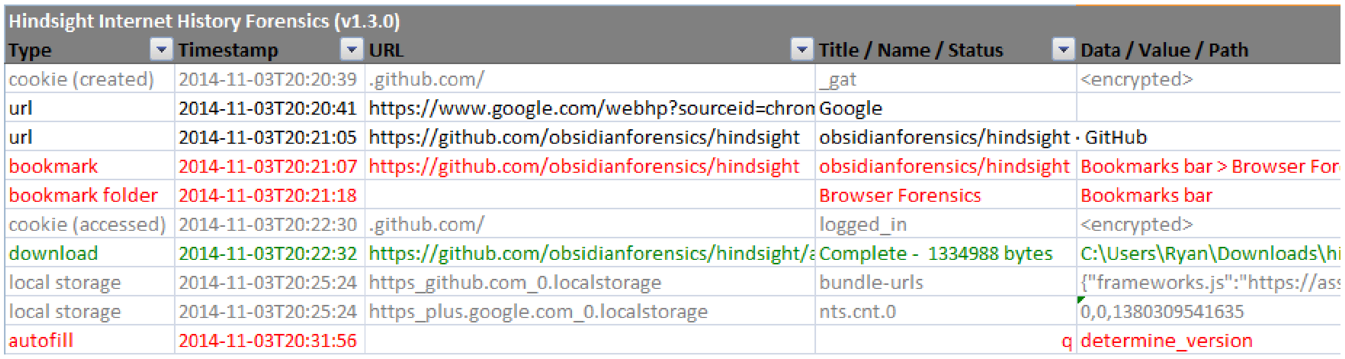
Figure 5: View of Hindsight 'Processing' and 'Plugins' sections

After all the artifacts are parsed, analysis using plugins begins. Hindsight will load all plugins located in the plugin folder. Each plugin has configuration information that lists key information about it, including its name, version, description, the artifacts it applies to, and if it uses remote resources. Hindsight checks this information to determine if the plugin should be allowed to run (for example, if a plugin uses remote resources and the `-r` option is not set, it will not run). Hindsight lists each plugin and the number of records it analyzed (if available).

Hindsight's last phase generates the report detailing its findings, in the format specified. Once Hindsight has finished creating the report, the finish time is displayed on the screen and the program exits. By default, the report will be in the same directory as `hindsight.py`, and the default XLSX formatted report can be viewed with a number of common spreadsheet applications.

Reading the XLSX Report

The main view in a Hindsight report is the 'Timeline' tab. This page is a timeline of all the record data that Hindsight was able to pull from the Chrome history files, sorted from oldest to newest. The record types are color coded to make it easier to digest at a glance.



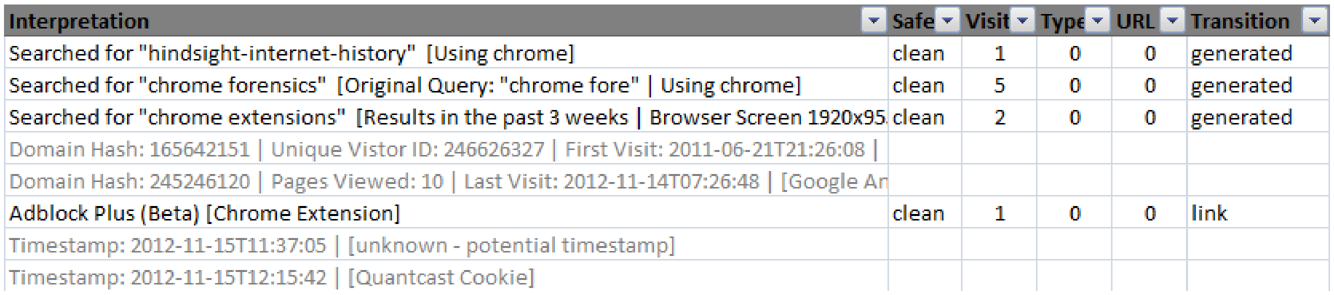
Hindsight Internet History Forensics (v1.3.0)

Type	Timestamp	URL	Title / Name / Status	Data / Value / Path
cookie (created)	2014-11-03T20:20:39	.github.com/	_gat	<encrypted>
url	2014-11-03T20:20:41	https://www.google.com/webhp?sourceid=chron	Google	
url	2014-11-03T20:21:05	https://github.com/obsidianforensics/hindsight	obsidianforensics/hindsight · GitHub	
bookmark	2014-11-03T20:21:07	https://github.com/obsidianforensics/hindsight	obsidianforensics/hindsight	Bookmarks bar > Browser For
bookmark folder	2014-11-03T20:21:18		Browser Forensics	Bookmarks bar
cookie (accessed)	2014-11-03T20:22:30	.github.com/	logged_in	<encrypted>
download	2014-11-03T20:22:32	https://github.com/obsidianforensics/hindsight/a	Complete - 1334988 bytes	C:\Users\Ryan\Downloads\hi
local storage	2014-11-03T20:25:24	https_github.com_0.localstorage	bundle-urls	{"frameworks.js":"https://as
local storage	2014-11-03T20:25:24	https_plus.google.com_0.localstorage	nts.cnt.0	0,0,1380309541635
autofill	2014-11-03T20:31:56		q	determine_version

Figure 6: Partial view of 'Timeline' tab

The first three columns, *Type*, *Timestamp*, and *URL*, are self-explanatory. Every row in the timeline will have type and timestamp values, and most will have a URL as well. The next two columns, *Title / Name / Status* and *Data / Value / Path*, are a little more complicated, as depending on the type of record they will contain different fields. The *Title / Name / Status* field generally describes the data in the *Data / Value / Path* field. For example, for Local Storage and Cookie records, the *Title / Name / Status* field has the name of the cookie, and the *Data / Value / Path* field has the cookie content. Autofill records are similar; the name of the input field is in *Title / Name / Status* and the entered value is in *Data / Value / Path*. The reason for collapsing these different fields into two is to make the data in the report more easily accessible. An investigator can scan down the timeline and see all the relevant information in one stream, rather than having to scroll across a dozen columns or switch to a different tab to view a different type of record.

The next column, *Interpretation*, is one of the key features of Hindsight. This is the primary place that plugins display their output. Each plugin processes a specific type of record and decodes the content to make it easier to understand. Plugins run the gamut from complicated to very simple, but regardless of their complexity each has the potential to save an investigator time by automating previously manual tasks.



Interpretation	Safe	Visit	Type	URL	Transition
Searched for "hindsight-internet-history" [Using chrome]	clean	1	0	0	generated
Searched for "chrome forensics" [Original Query: "chrome fore" | Using chrome]	clean	5	0	0	generated
Searched for "chrome extensions" [Results in the past 3 weeks | Browser Screen 1920x95	clean	2	0	0	generated
Domain Hash: 165642151 | Unique Vistor ID: 246626327 | First Visit: 2011-06-21T21:26:08 |					
Domain Hash: 245246120 | Pages Viewed: 10 | Last Visit: 2012-11-14T07:26:48 | [Google An					
Adblock Plus (Beta) [Chrome Extension]	clean	1	0	0	link
Timestamp: 2012-11-15T11:37:05 | [unknown - potential timestamp]					
Timestamp: 2012-11-15T12:15:42 | [Quantcast Cookie]					

Figure 7: Sample *Interpretation* column values

The next section of columns apply only to URL records. The first is labeled *Safe?* and shows how Google's Safe Browsing service classifies the URL (as malware, phishing, or clean); this column will be blank if remote lookups are not allowed or the plugin is not present. *Visit Count* gives the cumulative number of times that webpage was visited, and *Typed Count* shows the number of times that a user typed in the page's address (rather than clicking on a link). *URL Hidden* indicates (via a 0 or 1) whether the URL bar was visible to the user. *Transition* shows how the user arrived at the webpage (link, typed, start page, etc).

The last column grouping covers download-specific items. The *Interrupt Reason* column will either say "No Interrupt" or explain why a download was unsuccessful. *Danger Type* details any downloads that Chrome thought may be dangerous, and *Opened?* shows whether a user opened the completed download from within the Chrome interface. *ETag* and *Last Modified* both describe the downloaded file and are set by the hosting file server.

Chrome Artifacts

This last section provides a brief overview of the different types of Chrome artifacts that Hindsight can extract information from, as well as the artifacts' locations on disk and their file formats.

History

Location: <Chrome Dir>/History
File Format: SQLite
Chrome Versions: all

The 'History' file in the Chrome directory is the heart of where Chrome stores browsing records. This SQLite database has a number of different tables, but the two that combine to give most of the information about visited websites are the 'urls' and 'visits' tables. Hindsight extracts a number of fields for each website visited, including the URL, the page title, the visit count, the time the page was visited, and transition information. The 'History' database only keeps URL records for three months, but keeps download records with no time limit (download records explained below).

Archived History

Location: <Chrome Dir>/Archived History
File Format: SQLite
Chrome Versions: 1 - 36

The 'Archived History' file is similar to the 'History' file, but contains records that are over three months old. It has fewer tables than 'History', but the key 'urls' and 'visits' tables are still present and have the same structure. Hindsight extracts the same fields for these older records and places them on the 'Activity' timeline as `url (archived)`.

History Index

Location: <Chrome Dir>/History Index yyyy-mm
File Format: SQLite
Chrome Versions: 1 - 29

The 'History Index' files are another very useful artifact. For some sites Chrome records the text on the web page and saves it in one of these index files. There are four files, each covering a month, and are named 'History Index yyyy-mm' (with yyyy-mm designating the year-month, e.g. 2012-04). Along with the text data, Chrome records the page title, URL and the timestamp. Hindsight processes the index data and adds it to existing `url` and `url (archived)` records in the 'Activity' timeline in the *Indexed Data / Value / Path* column. Because the index information is timestamped, it can be possible to view the text of a web page at multiple points in time.

Bookmarks

Location: <Chrome Dir>/Bookmarks
File Format: JSON
Chrome Versions: all

Chrome stores its bookmark information in a JSON file in the root of the Chrome directory. From this file, Hindsight extracts the name of each bookmark, the bookmark's URL, the folder(s) the bookmark was saved in, and the date it was added. The tool also extracts when a bookmark folder was created and adds both these record types to the 'Activity' timeline.

Autofill

Location: <Chrome Dir>/Web Data
File Format: SQLite
Chrome Versions: 2 and on

The Chrome 'Web Data' file is a SQLite database with a plethora of valuable information. It has a number of interesting tables, but some of the most useful to an investigator are the ones relating to autofill data. Autofill is a feature of Chrome that is intended to help a user by remembering data that was filled into forms. When a user visits the same website again (or a different website with a similarly name input field), Chrome will automatically fill out the forms with the user's previous answers.

Hindsight extracts the name of the input field and the saved value, as well as the time that the value was used. No domain information is stored as to what website the autofill data was used on; however, by ordering the autofill and URL data by time it is easy to see what website the autofill data is likely associated with.

Downloads

Location: <Chrome Dir>/History
File Format: SQLite
Chrome Versions: all

Chrome stores records of files a user has downloaded in the 'downloads' table in the History file. Hindsight extracts the URL the file was downloaded from, the full path of where it was saved to locally, the number of bytes received vs. the total file size, and places this record on the 'Activity' timeline at the time the download started. Chrome changed the way it stores download information significantly in v26; it added the 'downloads_url_chains' table made changes to the 'downloads' table.

Cookies

Location: <Chrome Dir>/Cookies
File Format: SQLite
Chrome Versions: all

Unlike Internet Explorer, Chrome stores cookies in a SQLite database, not in individual text files. Hindsight reads this database and extracts each cookie's name and value, along with the website that placed it. Since Chrome records both the cookie's creation date and its last accessed date, Hindsight will create two entries per cookie in the 'Activity' timeline: `cookie (created)` and `cookie (accessed)`, but only when the timestamps are not identical. If the timestamps are the same, Hindsight will only display a `cookie (created)` row.

Websites use cookies to store information, often with the goal of "personalizing" users' experiences. This data can often be very useful to investigators. A number of Hindsight's plugins do some parsing of cookie data, ranging from simply translating Unix timestamps to decoding Google Analytics and Quantcast tracking cookies.

Chrome starting encrypting cookie values in the 'Cookies' database in v33. If conditions are right, Hindsight can decrypt these values (see http://www.obsidianforensics.com/blog/hindsight-v1-2-0-released-adds-cookie-decryption-and-logging/ for more details).

Local Storage

Location: <Chrome Dir>/Local Storage/http_www.example.com_0.localstorage
File Format: SQLite
Chrome Versions: 5 and on

Local Storage is a common name for part of HTML5 Web Storage. Local Storage is the newest version of cookies, and it serves the same purpose as “normal” cookies: enabling websites to store persistent data locally. This new iteration of cookies is superior in many ways, including increasing the amount of data each site can store (from around 4KB in old HTTP cookies to about 10MB in HTML5 Local Storage).

Chrome implements Local Storage by creating a .localstorage file in the ‘Local Storage’ directory for each website that elects to use it. Each .localstorage file is a SQLite database that holds all the key/value pairs. Because no temporal information is stored by default in the database, Hindsight uses the last accessed time of the .localstorage file itself to place the Local Storage records on the ‘Activity’ timeline. However, since the website can store whatever type of information it wants, there is the chance that relevant timestamps could be stored in the content of the database. If there are multiple key/value pairs in a website’s .localstorage file, Hindsight creates a separate entry for each one in the timeline, all with the last accessed time of the file itself.