2009 Eighth IEEE International Symposium on Network Computing and Applications

FIRST: Forensic Internet Replay Sequencing Tool

Michael D. Kelly and Sean J. Geoghegan
Computer Science Department
University of Arkansas at Little Rock
Little Rock, AR USA

Abstract* – This paper discusses the design and development of a tool that can be used by digital forensic investigators to display a visual recreation of web browsing sessions based on the browser cache stored on a suspect's hard drive. Frequently, a forensic investigator will search a web browser cache to learn about the browsing habits of a suspect. The data gathered from the cache will be used to support a case in a civil or criminal proceeding. Thus, the investigator must have a method to clearly convey the information learned from the investigation of the cache. Currently, tools exist that can parse the browser cache to determine the Uniform Resource Locaters (URLs) visited by the suspect and the times that the URLs were visited. While this data indicates the browsing habits of the suspect, a list of URLs and timestamps is not easily understood by non-information technology professionals. The goal is to construct a tool that will assemble a visual slideshow that recreates, as closely as possible, a suspect's browsing experience. This recreation will help provide investigators and other interested parties with an accurate understanding of the suspect's intentions while browsing the web.

Keywords- web browser, digital forensics, cache, URL

I. INTRODUCTION

Many investigations involve creating a timeline of the activities that occurred leading up to the event in question. This timeline is used to determine the location and activities of the individuals involved in the event. As the Internet has become more prevalent in society, more human activities are taking place online. Therefore, evidence of illegal activities is increasingly available by investigating the tools used by a suspect to access on-line information sources. Web browsers such as Internet Explorer have a built-in caching feature to improve performance while a user is browsing. When a user accesses a website, the data that is downloaded is stored in the cache and used to render the webpage in the browser. If the user visits the same website, the data stored in the cache is used to render the webpage, rather than re-downloading the data [1]. Since the files are stored in the cache, repeated visits to the same page result in faster viewing. This cache has become an important source of information for forensic investigators because it contains files from a suspect's browsing history that may be several days or even weeks old. The investigator uses this data to ascertain the websites visited by the suspect, the time that the suspect visited the sites, and the information that was available to the suspect. For example, an investigator may find evidence that a suspect in a murder case visited several websites related to a poison that was found in the victim's blood. This evidence may help convince a jury that the suspect knowingly and premeditatedly murdered the victim.

This work is supported, in part, by research supported by the Arkansas Department of Higher Education. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

Currently, law enforcement investigators utilize tools that read the indexes of the browser cache to create a list of the URLs that were accessed. This list includes information such as the time that the URL was accessed, the directory that the file is stored in, and the HTTP headers. These tools are effective in creating a text-based timeline for the browsing history, which is invaluable to an investigator who is trying to determine the events leading up to a crime. However, presentation of data in this format to a jury of non-technical individuals is often cumbersome and ineffective. The impact of the suspect's browsing history is often lost when it is reduced to a simple textual display.

An effective enhancement of the textual representation of the history is the construction of a visual recreation of the browsing history. This visual recreation will be much easier for jury members to understand since it closely matches an individual's web experience. A textual display is not familiar to the typical users and must be explained in detail by the forensic expert. In contrast, the visual recreation is much more intuitive. For example, showing a list of URLs and timestamps of websites related to poisoning deaths is much less effective than showing the jury the content of the websites.

A type of visual representation that is very useful to law enforcement is a slideshow. Investigators must demonstrate their findings in an arbitration setting where the audience may not have technical knowledge. A slideshow is an effective means of communicating the activities of an individual to such an audience. The slideshow will show the audience the content and order that websites were visited in a manner that is similar to the original viewing by the suspect.

II. RELATED WORK

There are currently programs that function similarly to FIRST. IE History & Cache Viewer can be found at the privacywindows.com website. Web Cache Illuminator can be found at the nstarsolutions.com website. They provide a list of all items found in the cache, an information area about each item, and a viewer window that lets you look at each item. However, there are some limitations that prevent them from being useful for forensic purposes. First, they must be installed on the target computer. This requirement is a significant disadvantage because by modifying the target computer, the installation may destroy the integrity of the evidence. Another disadvantage is that they do not discriminate the items in the cache, they merely display them all. The viewers also only load the single item that is chosen. If you have a static webpage with a single graphic, it will show up in the software as an HTML file and an image file. When you click on the HTML file, only

978-0-7695-3698-9/09 $25.00 © 2009 IEEE
DOI 10.1109/NCA.2009.44

the text information in that file is displayed. The corresponding image is not displayed within the viewer. FIRST was designed to address these limitations.

III. METHODOLOGY

A. URL Discrimination

The first requirement for the project was choosing the indexing parser software. A preliminary investigation into the available tools showed that the amount of output differs with each tool. Some tools list the URL for every element on the webpage, while others seem to list the webpages themselves. The index parsing software that was chosen needed to deliver a clean, field-separated output that can be parsed again for conversion to slideshow input.

The purpose of discriminating data is to separate wanted files from unwanted files in order to reduce the amount of data processed [2]. In this case, discriminating between a URL that represents a webpage and a URL that represents an element on a webpage is crucial for determining how to reconstruct the browsing session. A tool that lists every element as a separate URL is not useful for this application as it is difficult to determine how the URL was accessed. For instance, an image file may be accessed as an element of a webpage, such as a corporate logo, or be accessed by the user directly, such as when looking at software screenshots. Software was needed that could provide output that lists only those URLs that represent an entire webpage, even if that webpage consists of a single image file.

In order to find the tool that provided the most useful output, it was necessary to perform a comparative analysis of the outputs from various forensic Internet history tools. A known set of URLs was accessed from a computer with no previous browsing history in order to provide a control for comparison. Each tool was used to analyze the browser history and output the results. The output was then examined to determine which tools generate useful output. A preliminary investigation showed that the Pasco software can examine Internet Explorer activity and successfully output a sorted listing of webpage URLs [1]. Further testing using the controlled system described above determined that Pasco properly discriminates between URLs and has the necessary licensing in place to allow its inclusion in the FIRST project.

B. Translating tool output to slideshow input

The output of the software tool needs to be directed to a program that can display the individual URLs in time-sorted order. The best choice for this is a web browser. At this time, web browser technology progresses fairly rapidly. If a custom display browser is built, this progress will cause the browser to become outdated and unequipped to handle new ways of displaying information. Maintaining the custom browser to keep it from becoming obsolete will become an enormous task. It will also be difficult to ensure that the browser renders webpages identically to Internet Explorer, unless the rendering engine is the same. Also, a forensic investigator will benefit from viewing the webpages in the exact same manner that the suspect viewed them. Additionally, it is beyond the scope of this project to recreate the functionality of Internet Explorer. This leads to the use of Internet Explorer itself to display the URLs that are output by the index parsing tool.

A significant concern in loading the URLs is making sure that the webpages and files that they point to are loaded from Internet Explorer's cache instead of retrieved from their respective web servers. The ability to load webpages from the cache addresses the issue of whether the content was updated or deleted. Loading the files from the cache will ensure that the investigator will see the webpages exactly as the individual saw them. Internet Explorer has a function for offline browsing that uses the cache to display pages when an Internet connection is unavailable. Initial testing indicates that this feature is able to display webpages from the cache that are visually identical to the webpage as it was originally viewed. This is not surprising since the files that compose the webpage are the same files that were retrieved from the web server.

Since Internet Explorer will be used to display the images, the output from the parsing tool must be translated into a format that Internet Explorer can display. The obvious choice for this format is HTML. HTML has features that allow two methods for displaying the slideshow. The first method would be to create a two part viewer webpage with the URL listing on one side and an area to load the webpages on the other. This method will allow the investigator to view the webpages in any order at will. The disadvantage of this method is that what is shown on the screen is not identical to what the suspect viewed.

The other method would be to use HTML redirect meta-tags to automatically load the next website after a specified number of seconds has transpired. The advantage of this format is that it allows a correspondence between the slide show and the original viewing time. The disadvantages are that the investigator cannot browse the pages at will, the webpages must be viewed in order, and any large delays between webpages must be endured on each view. Large delays may be shortened by giving a maximum time between webpages, but this is done at the expense of a faithful reproduction.

The FIRST program was constructed using Microsoft's .Net programming language. This allowed the easy use of .Net's WebBrowser control, which is an embeddable version of Internet Explorer. Use of this control allowed FIRST to act as a wrapper around Internet Explorer. This wrapper is much like a customized bookmarking system with automated advancement through the URLs. This gives the advantage of using the same rendering engine as Internet Explorer [3] as well as eliminating the need to alter the contents of the cache by adding redirect commands, which compromises the integrity of the cache evidence. There is still the disadvantage of having what the suspect saw compressed into a smaller viewing area inside another program, but the displayed information is identical to what the suspect saw.

FIRST reads the URLs and other information given by Pasco, sorts them into chronological order, and places them into a combo-box that lets the user view the order of display and freely choose among them. Display controls much like those found on a DVD player are also present to allow the user to play, pause, fast forward, and rewind the automated playback. Finally, there is also a printing feature to allow physical copies of the displayed webpages to be created.

C. URL Discrimination

Testing experiments were run in order to determine Internet Explorer's caching behavior and the limitations of the cache. For each test, the computer was cleared of all Internet activity and the browser initially opened to a blank page. Except where otherwise stated, all web pages were accessed by typing the URL directly into the address bar. The computer used for the experiments is a Dell Optiplex GX-60 running Windows XP Professional, Version 2002, Service Pack 2. Microsoft's Internet Explorer 7 was used for the web browser.

A suite of web pages was chosen for the type of content that each contains. One was a static webpage with basic text and images. Another contained JavaScript, while a third had both JavaScript and CSS. The fourth contained client-side scripts. These four sites were a good representation of various Web technologies that are commonly encountered on the Internet.

The cache itself consists of the History and Temporary Internet Files (TIF) folders located in each user's folder. The TIF folder holds four subfolders that contain the actual files downloaded from the Internet. These subfolders are given random names by Internet Explorer for security reasons. [5] A file called index.dat contains the locations of all the parts of each web page that is downloaded.

Various tests were run to determine the behavior of the cache. This was necessary to find out how to discriminate URLs in order to automate the process. By loading webpages in the browser and examining the contents of the cache, a picture began to form about the possibilities and limitations of what could be recovered.

One test showed that the separate files making up a webpage are not necessarily saved to the same cache subfolder. In addition, HTML files are renamed to reflect the domain or path on the domain from which they were retrieved. Another test showed that the favicons, tiny icons that are displayed in the address bar, are saved in each of the folders with a number added to the filename. Once the folders are full, the number gets incremented. Since parts of the webpage that were known to be hidden until a button was pushed were found in the cache, it was determined that the entire webpage is downloaded on first access.

Other tests showed the effects that updating a webpage and use of the Back and Forward buttons have on the cache. Two webpages were loaded into the browser in the order: page one, page two, page one. Page one only showed up once in the cache and showed the same using the Back and Forward buttons. Page one was updated and loaded after the previous entries. The cache then reflected only the updated page.

Experiments involving both server-side and client-side scripting were performed. A search engine using server-side scripting was queried five times to overflow the cache subfolders. As with the favicons, the search result pages were all saved with the same name in different folders. In the case of the fifth page, it was in one of the folders with the trailing number incremented. This test showed that server-side scripting is most likely preserved by the cache.

Client-side scripting was shown to be very problematic for forensic work. The state of a client-side script is not preserved even within the browser using the Back and Forward buttons. The scripted elements return to the original state. In the cache, these elements are renamed and scattered through the subfolders. In order to restore the webpage, they must be found, renamed to the original name, and placed in the same folder as the HTML file. This causes the cached to be tampered with, which is not conducive to a forensic investigation. Also not conducive is the fact that the suspect's interaction with the script is lost, including any popup boxes, dialog boxes, or alerts, which are all system calls that do not get saved to the cache.

In one test, a webpage with a randomly chosen image was loaded into the browser and refreshed until the image changed. All network access was removed from the machine and only the second image was accessible in the browser. Both images and two copies of the webpage were found in the cache in different folders. Clicking on both copies of the HTML file resulted in plain text pages with no image being loaded into the web browser. Examination of the source code revealed that the graphic features of the page, including the pictures being tracked, are pulled at display time from other areas of the UALR network. Plugging the network cable back in and refreshing the pages from the saved cache on the secondary media resulted in some of the fonts and images being restored, but not the images that were being tracked. At this point, the only way to know that the image belonged to the webpage is to go through the Cascading Style Sheet (CSS) file describing the webpage.

A simple web page was loaded into the browser twice. The page was then altered and loaded again.. The second instance of the webpage was refreshed and it showed the altered version. Clicking on the back button displayed the altered page again, just like in the previous experiment. The web page was returned to the original, unaltered version and uploaded to the server again. The browser's cache was cleared and the web page was loaded. The original page was displayed in the browser. Once again, the simple page on the server was altered and loaded into the browser. At this point, the pages were loaded as follows: original, then altered. The network cable was unplugged and the back button was clicked to return to the original. What should have been the original document was the altered version. Examining the cache revealed that only the altered version of the web page was saved. This experiment shows that even if network connections are severed, only the latest version of a web page is saved in the cache and available for retrieval and display.

A third experiment was performed in order to establish the effects of AJAX technology on the cache. AJAX technology allows a webpage to change with user input by reloading parts of the page as needed. Only the part of the page that contains the new information is updated.

Inside the browser, navigation is broken when using this technology. Since only part of the webpage is changed, the Forward and Back buttons have no control over navigating different versions of that page. Inside the cache, the changed parts of the webpage are stored as separate files that contain no easily determined identifying information. The filenames appear to be assigned randomly, but further testing will need to be done to confirm this. A timestamp could be used to place the files in a sequence to show that changes were made, but the files are not displayable and do not show which changes are made without examining the code within.

Some interesting effects were observed during this experiment. Upon loading an AJAX webmail site, a link to an email folder was clicked in order to observe which parts changed. Network access was removed from the system and an individual email link was clicked. The email loaded up like normal even though there was no access to any network, showing that some parts of the page are prefetched from the server. There also seems to be an expiration time on the changes. After about 10 seconds, the prefetched links no longer worked and only displayed error messages. After restoring network access, the links worked again, but the content was downloaded again. This is not a problem for things like an email, which does not change over time, but will cause forensic recovery issues if used on a page that contains frequently updated material.

D. *Testing First*

The cache used in the above experiments was saved to a flash drive and stored during the design and programming of FIRST, which took several months. The significance of this is that the webpages in the cache had time to age past any expiration dates set by the browser. These expiration dates are used by Internet Explorer to determine if it should try to check the server for an updated page. [4] If the expiration date has passed, then the page is fetched from the server again and the file in the cache is overwritten.

In order for FIRST to function properly, and to ensure that only the files in the cache are used, certain conditions must be met. All connections to the Internet must be severed. The preferred method of this is to disconnect any network cables, turn off wireless adapters, and activate Offline mode in Internet Explorer before running FIRST.

There are two ways that a forensic tool like FIRST can damage the integrity of the cache evidence. The first way is to require installation to the computer. FIRST was designed to not have an installation requirement, so it can be run from external media. The second way is to alter the forensic data. Windows will not recognize a cache that is in an alternate location. It can see the files, but does not treat it the same as Internet Explorer's cache. The workaround for this limitation is to load a forensic copy of the user's profile to the computer, log into that user account and run FIRST from there.

Once a machine was set up correctly for FIRST as described above, testing proved to be reasonably successful. Webpages in the cache were found and loaded into FIRST and the pages were displayed in 10-second intervals. Most pages were displayed with all elements intact. The limitations found were difficulties in reproducing AJAX pages, client-side scripting, displaying Cascading Style Sheets, and revisiting a page.

IV. CONCLUSIONS

Forensic investigators have a need for a program that can read the contents of the browser cache and display them in the same format as the suspect viewed them without destroying the integrity of the cache evidence. Current tools provide only a list of URLs for each of the elements in the cache. Some programs let you view each element, but they do not recreate the entire webpage as it was viewed by a suspect. FIRST provides a level of discrimination that limits the list of URLs to only those that represent full webpages, not the single elements. Most of those URLs can then be viewed within FIRST exactly as the original viewer saw them. This is a valuable tool that lets an investigator provide a jury or other interested party with a first-hand experience of a suspect's Internet activity.

V. FUTURE WORK

There are some issues that need to be addressed in future work:

1. Client-side scripting interaction is lost. We can recover the page that contains the "widget", but any interaction with it is lost.
2. Only the latest version of a webpage is preserved in the cache. Earlier versions are overwritten and lost.
3. Cascading Style Sheets create a problem for recreating the page correctly. The CSS code must be examined for the proper directory structure in order for everything to display correctly.
4. AJAX technology causes breakage of the navigation buttons and history. The webpage is fragmented into many files and updated as needed. There is some prefetching of data, but it expires quickly and becomes unavailable.
5. A method for programmatically setting the operating system to Offline mode and loading an alternative cache location would be very helpful to examine the cache without impacting the integrity of the evidence.

REFERENCES

[1] Keith J. Jones, "Forensic Analysis of Internet Explorer Activity Files", revised May 6, 2003. Retrieved from http://easynews.dl.sourceforge.net/sourceforge/odessa/IE_Internet_Activity_Reconstruction.pdf on October 18, 2006. The Pasco software can be found at http://sourceforge.net/project/showfiles.php?group_id=78332 and is released under the BSD license.

[2] Bill Nelson, Amelia Phillips, Frank Enfinger, Christopher Steuart, "Guide to Computer Forensics and Investigations 2nd Edition," Thomson Course Technology, 2006.

[3] "How To: Add Web Browser Capabilities to a Windows Forms Application". Retrieved from http://msdn2.microsoft.com/en-us/library/3s8ys666(vs.80).aspx on April 19, 2007.

[4] Wolfgang Baudisch, "Off-line issues: Cannot Find Server", revised December 7, 2004. Retrieved from http://www.wbaudisch.de/Offline.htm on February 2, 2007.

[5] Sandy Hardmeier "General Information about the IE Cache", 2006. Retrieved from http://inetexplorer.mvps.org/answers/57.html