Digital Investigation 14 (2015) 63–75

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin

BrowStEx: A tool to aggregate browser storage artifacts for forensic analysis

Abner Mendoza [a], Avinash Kumar [b], David Midcap [b], Hyuk Cho [b], Cihan Varol [b, *]

[a] Department of Computer Science and Engineering, Texas A&M University, College Station, TX, USA
[b] Department of Computer Science, Sam Houston State University, Huntsville, TX, USA

ARTICLE INFO

Article history:
Received 7 November 2014
Received in revised form 25 July 2015
Accepted 1 August 2015
Available online 28 August 2015

Keywords:
HTML5
Local storage
Persistent storage
Web browser forensics
Web storage

ABSTRACT

Web storage or browser storage, a new client-side data storage feature, was recommended as a part of the HTML5 specifications and now widely adopted by major web browser vendors. Web storage with native browser support has changed the paradigm of web application development unprecedentedly because persistent data storage with increased data size can be realized on the client. Web storage is poised to quickly become an area of particular interest for forensic investigators due to the potential to discover critical information from web browser artifacts at client side. However, the literature work on web browser forensics has traditionally focused on browsing history, browser cache, and cookie files (Oh et al., 2011). Therefore, we first discuss the prevalence of web storage implementation in widely used websites. Then, we compare and contrast the web storage technology currently implemented in the five major web browsers, Google Chrome, Internet Explorer, Mozilla Firefox, Opera, and Apple's Safari. Moreover, in order to provide more insights into web storage and enable unified forensic analysis, a proof-of-concept tool, named as BrowStEx (Browser Storage Extractor), is described with implementation details. The commonalities, differences, and the proof-of-concept tool discussed in this paper can be useful in developing advanced forensic tools that can extract browser storage artifacts.

Introduction

With the increasing reliance on the web to sustain a modern lifestyle, a wealth of information can be harvested by analyzing a user's web browsing activities. Indeed, the bulk of a user's interactions with end-user workstations today is related to Internet communications (Altheide and Carvey, 2011). Accordingly, web browser forensics is fast becoming an important investigation topic within the computer forensics field. As web technologies evolve and generate explosive amounts of data, the methods and tools for extracting useful information need to properly handle the volume, the velocity, and the variety of data accordingly. Computing devices are becoming more cloud-based, with requirements to work seamlessly with and without an Internet connection; thus, web developers have to cope with this paradigm shift by utilizing HTML5 (HyperText Markup Language 5) APIs (Application Programming Interfaces) that support persistent data storage even when devices are offline. The most popular of these persistent data storage mechanisms is a new HTML5 API known as Web storage, sometimes also referred to as Browser Storage, HTML5 Storage, Local Storage, Offline Storage, or DOM (Document Object Model) Storage (Laine, 2012). For consistency, we refer to these collectively as Web storage in this paper. Web storage is a browser-based API bundled in

* Corresponding author.
E-mail address: cvarol@shsu.edu (C. Varol).

http://dx.doi.org/10.1016/j.diin.2015.08.001
1742-2876/

HTML5 specifications that allow persistent client-side storage for web applications. As major web browser vendors adopted this new storage mechanism, developers have realized the benefits of being able to store persistent data on the client side. This new mechanism for client side data storage is appealing not only because it is natively supported by browsers, but also because it offers much larger storage capacity than that was available with older mechanisms such as cookies. Additionally, unlike cookies, web storage data does not get transmitted with each request to and from a web server, thereby reducing the bandwidth overhead. Despite the increased use of web storage among the popular websites today, to the best of our knowledge, there is almost no detailed discussion on this newly adopted technology in browser forensic investigation literature. Therefore, this paper intends to discuss the prevalence of web storage implementation in current major web browsers, emphasize the high potential of finding information on client-side web storage data generated by multiple web browsers, and guiding in developing web storage forensics tools.

We first compare client storage mechanisms and provide the characteristics of HTML5 web storage. Then we provide an evidence of how prevalently the web storage technology has been adopted by web browsers. Particularly, we compare and contrast the details and nuances of web storage implementation among the five major web browsers, Chrome, Explorer, Firefox, Opera, and Safari. Furthermore, in order to provide more insights and guides in developing web storage forensic tools, a proof-of-concept tool, named as BrowStEx (Browser Storage Extractor), is described. BrowStEx can extract the data artifacts from related web storage files and present them in an aggregated view for unified analyses.

The remainder of the paper is organized as follows. In Section Prior work, we discuss related work about web storage. In Sections Persistent storage and HTML5 web storage, the details of persistent storage and HTML5 client-side storage in different browsers are summarized. The findings on web storage implementation and stored data are discussed in Section Web storage implementation and utilization. In Section BrowStEx: design and implementation, the BrowStEx application is introduced. Finally, we conclude the paper with summary and brief discussion on future research direction in Section 7.

Prior work

Other than the W3C (World Wide Web Consortium) draft specification guidance (W3C, 2013), browser vendors have not disclosed much details regarding the implementation of web storage technology in their respective browsers. Moreover, there seems to be no comprehensive research that compares and contrasts details on how web storage is implemented and incorporated among different browsers. Prior research work related to HTML5 web storage has mostly focused on the ways in which web storage may be used, such as the variety of stored data, and specifically, the issues surrounding security and privacy.

West and Pulimood (2012) provided an analysis of the web storage specification and its usage in the context of privacy and security issues. The authors implemented a simple budget management web application, which uses web storage to store data locally on the client machine and periodically synchronizes the data with the server. Their experimental study provides an insight into how improper usage of web storage could expose sensitive data to other users using the same web application on the same system, such as users in an Internet cafe. The study exemplifies a potential security breach of personal data and also hints at an opportunity in the context of digital forensics. For example, if a website records search queries, browsing history, and other user information into web storage, it could provide invaluable information in a digital investigation, especially if a user has attempted to hide his/her browsing activities by clearing the browsing history simply using the browser-provided history deletion functionality. In such a case, the web storage contents would be a better target for investigation.

Silo (Mickens, 2010) leveraged web storage to cache JavaScript and CSS (Cascading Style Sheets) chunks on the client side, thereby improving the performance of web applications by requiring less data transmission between the browser and the remote web server. Bogaard et al. (2011) explored how malicious web developers could exploit web storage and use it as a covert channel to distribute sensitive information across the Internet and also to retrieve it at a later time. The premise of this technique is that if a malicious user does not want to store information locally in their own machine or in a server, they can store it in unsuspecting client machines leaving no evidence of the data in their own machine. The authors built a web application that would break a file into 26 portions and distribute each portion to a large number of clients on the Internet that visit the web application. The goal of the experiment was to show that the file could be reconstructed eventually and to investigate how many copies of each portion of the file were necessary to be distributed to increase the probability that it could be restored on subsequent visits. The authors exploited the fact that web storage usage by any given web application is often transparent to the user, while that the process for manually removing web storage data is not very intuitive to the user. They also highlighted that almost any type of data could be stored in web storage simply by changing it to a text-based representation that could be later converted to its native format. This means it is difficult to prevent a malware distributor from using this storage to store malware payload, or perhaps code that could be injected into the browser to download malware. Similarly, a nefarious developer could easily use a binary-to-text translation tool to store binary files in a client machine. Lekies and Johns (2012) showed how malicious users might use web storage to inject JavaScript payload files into web applications. The authors provided an excellent overview of three attack scenarios, which shed light on the vulnerabilities associated with web storage. In their research, they investigated the top 500,000 domains, as ranked by Alexa (2013), and showed that web storage was already widely implemented, especially among the most popular websites.

These previous studies show the versatile use of web storage. Current trends indicate that web storage is mostly