# Acquisition of Evidence of WebStorage in HTML5 Web Browsers from Memory Image

Shinichi Matsumoto*†, Kouichi Sakurai*†
*Institute of Systems, Information Technologies and Nanotechnologies (ISIT)
Momochihama, Sawara-ku, Fukuoka, Japan
†Department of Computer Science and Communication Engineering, Kyushu University
Motooka, Nishi-ku, Fukuoka, Japan
Email: smatsumoto@isit.or.jp, sakurai@inf.kyushu-u.ac.jp

***Abstract*—Web browser is a growing platform for the execution of various applications. There are large fractions of smartphone platforms that support the execution of web technology based application, especially one such as HTML 5. However there are also some emerging smartphone platforms that only support web technology based applications.**

**Taking into the considerations of these situations may lead to a higher importance of forensic investigations on artifacts within the web browser bringing about the usefulness of the HTML5 specific attributes as evidences in mobile forensics. Through this paper, we explore the results of experiments that acquire the main memory image within terminal and extract the webStorage data as an evidence of the browsing activity.**

**The memory forensics of web browsing activity is highly concerned. The evidences gathered from the HTML5 webStorage contents acquired from the main memory image are examined and the results of the observations indicate the ability to retrieve webStorage from the memory image is certain. Therefore, we proclaimed formats of evidences that are retrievable from the main memory. The formats were different depending on the type of web browser accessed.**

**Three most utilized web browsers are experimented in this paper namely, Google Chrome, Mozilla Firefox and Microsoft Internet Explorer. The results showed that the acquisition of webStorage content on the browsers were possible and elucidated its formats. Values of webStorage is contained in the residuals that left by all of three web browsers. Therefore, if the investigator has the knowledge of values, he will be able to find the location of the evidence to hint values. If the investigator does not have the knowledge about the value, then he can explore the evidence based on the knowledge of the origin or key. Because the format of the evidence depends on Web browser, investigator must use different search techniques according to the Web browser.**

***Keywords*-Digital Forensics; Memory Forensics; Data Privacy;**

## I. Introduction

### A. Background

The importance of digital forensics are increasing from time to time and are publicly well recognized. In the digital forensics field, it is important to preserve the information on the activities of companies, organizations or individuals. And these information will be disclosed in an understandable and accurate manner.

This work was supported by JSPS KAKENHI Grant Number 26330169.

The need of digital forensics is growing not only for the purpose of criminal trials, but also for the civil trials. Specifically focusing on the e-Discovery, discovery in civil trials. Looking at the trend of digital forensics, it is expected that the importance of evidence of intrusion, accounting information and personal/confidential information will increas in the future.

Sammons[1] describes the digital forensics that can be utilized in various domains.

Criminal Investigation
: Nowadays, digital forensics does not covers only crimes which utilizes networks and digital equipments. Digital evidence can be left and found "analog" crimes can leave electronic evidence.

Civil Litigation
: In civil cases, both parties need to submit evidences in accordance with legal processes (discovery). Evidence treated in discovery has been changed from paper based evidences to digital evidence. The market of digital forensics in civil cases grows to become a huge business these days.

Intelligence
: A diplomatic and/or anti-terrorism activity utilizes digital evidence. Collection of communication records by division involved in national security has evolved to become a big scandal in recent years.

Administrative Matters
: Violations of corporate policies and procedures often leave digital evidence. These activities may not conflict with the law, but it may be a problem to the organizational administration. Collection of evidences related to these activities has become essential in terms of corporate governance.

### B. Motivation

The use case for digital forensics on mobile devices exists and it's demand increases as time advances.

As described in the previous section, the mainstream of mobile device is shifting from featured phone to smartphone. Execution environment for installable applications is one of the most defining features of the smartphone. Application execution environments are specific to the smartphone platform. For example, Android OS equips Dalvik Virtual

978-1-4799-5733-0/14 $31.00 © 2014 IEEE
DOI 10.1109/AsiaJCIS.2014.30

Machine in its framework that enables the execution of the Applications written in Dalvik bytecode. Android framework enabling application execution equips an application with the verification function, and verifies the Dalvik bytecode, application manifest. On the other hand, iOS and Tizen equips similar mechanisms for application verification and execution.

Although, the smartphone platforms equip other application platform, these platforms has web framework that enable the execution of web technology based application. Such application is described with HTML5, JavaScript and CSS (Cascading Style Sheet). Specifically in the case where Firefox OS supports only web technology based applications. This is the consequence of considering the portability as an aspect between platforms. We are fairly certain that the number of web technology based applications will increase.

Web technology based application will spread for smartphone platforms and also personal computer and home appliances. In this situation, the necessity of forensic technologies for HTML5 artifact is highly needed.

### C. Forensics for HTML5 artifacts

HTML5 specification defines numerous new functions to help web application development. Described in Section III, the APIs for handling multimedia data directly, and webStorage function enables to record variety of information on the client side (Web browser). Information stored in webStorage is defined by application, for example.

- In the mail application, header, from/to address, body contents of mails and attachments.
- In the photo album application, photo image data and some metadata.
- In the office application, documents and document editing history.
- In the game application, account information and play records.

For the application of forensics in mobile devices, targeting the webStorage increases information to be handled significantly.

### D. Related Works

Researches on digital forensics that target the web browser [2] [3], treats log files generated by web browser and SQLite database that stores internal information of web browsers were carried out.

On the other hand, Donny [4] attempts to retrieve the evidences on private browsing mode of various web browsers (Mozilla Firefox, Google Chrome, Microsoft Internet Explorer, Apple, Safari).

- MS Internet Explorer: InPrivate Browsing
- Google Chrome: Incognito mode
- Mozilla Firefox: Private Browsing
- Apple Safari: Private Browsing

Furthermore, they experimented on portable browsers below.

- Mozilla Firefox Portable
- Google Chrome Portable
- Opera Portable

Mulazzani [5] and Satvat [6] attempted to acquire evidences on the private browsing mode of web browsers, too. Furthermore, Aditya [7] experiments memory forensics on evidence left by private browsing mode of Web browsers.

Satvat [6] examined comprehensive attack surface on private browsing mode including DNS, Memory inspection, file timestamps, Index.dat file, extension, and so on. They revealed much vulnerability in the private browsing mode of web browsers.

They mainly acquired and analyzed the main memory image. Type of evidences that was able to be acquired successfully depended on the browser, but the successful ones are listed below:

- Indicator that describes pages has been browsed in private browsing mode.
- browsing history
- image image data
- username / e-mail account

In the meanwhile, Aggarwal [8] analyzed mechanisms of private browsing modes in modern browsers. They targeted on the latest Web standards HTML5. Especially in the behavior of localStorage in private browsing mode in Web browsers.

### E. Challenging Issue

We have to investigate whether or not evidence of web browsing activities is left in volatile main memory from the viewpoint of webStorage. WebStorage consists of three parts, the origin, key and value. Retrieving all of them is the desire of every investigator.

Retrieving them from the main memory is desirable for investigator. But as far as our knowledge, the literature on this problem has not been issued.

Furthermore, modern computer system comes with several gigabytes of RAM. Dumped memory can often be the several gigabytes. Searching evidences from large sizes of dumped file is difficult. Evidence string can be searchable if the string is known completely. Investigators can search and match the exact string from the evidence. If not so, searching with short fragment of string is difficult and takes a longer time.

Even if evidence can be observed, format and character code is not clear now. To investigate the memory forensics of webStorages, it is extremely important to be clear of what kind of evidence in the form of format and code of evidences is retrievable.

### F. Contributions and Result

In this paper, we are concerned with the memory forensics of web browsing activity. It is elucidated from the result of experiments conducted. Experiments were conducted to obtain the main memory image after web browsing activity and attempts to retrieve the evidence of webStorage contents were carried out. Experiments were conducted in the same manner as the three major types of web browser (Google Chrome, Mozilla Firefox, Microsoft Internet Explorer). As a result, it

is revealed that the contents and the format of webStorage can be retrieved from memory image.

Summary of the result is shown in I. In this table, N/A in the table indicates any evidence that could not be retrieved. Meanwhile, checked box indicates some form of evidence that can be retrieved. Forms of evidence left by each browsers are described in detail in IV-E.

TABLE I
ABSTRACT OF EXPERIMENT RESULT

| Browser | storage type | origin | key | value |
|---|---|---|---|---|
| Mozilla Firefox | localStorage | ✓ | ✓ | ✓ |
| | sessionStorage | ✓ | ✓ | ✓ |
| MS Internet Explorer | localStorage | N/A | ✓ | ✓ |
| | sessionStorage | N/A | ✓ | ✓ |
| Google Chrome | localStorage | N/A | ✓ | ✓ |
| | sessionStorage | N/A | ✓ | ✓ |

### G. Comparison with Existing Work

Jansen [9], Willassen[10] and Ahmed [11] are discussing about forensics techniques on featured phone. They treated the techniques to retrieve evidence from non-volatile memory within featured phone.

There many researches on private browsing mode of web browsers. Donny [4] make full use of memory forensics experiments to investigate the evidence left by private mode of Web browsers. Furthermore, Aditya [7], Mulazzani [5] and Satvat [6] utilized memory forensics and other techniques.

But these researches focused on the evidences related to HTML 4. As far to our knowledge, evidence related to HTML5 intrinsic APIs are not proven. We experimented memory forensics on the evidence left by HTML 5 webStorage.

## II. MEMORY FORENSICS

Memory forensics is growing and occupies major techniques in computer forensics. Traditional computer forensics treats persistent memory within computer system, such as Hard Drives, Flash Memory. Meanwhile, memory forensics captures the image of volatile main memory. The data contained in main memory is assumed to be lost easily such as in the end of the process and reboot of the terminal.

Importance of memory forensics is comprehensively discussed [12]. Residuals of browsing activities remains in main memory [13].

## III. ABSTRACT OF HTML5

HTML5 is the achievement of the development on web technology to be the platform to execute application.

It is a standard of web markup language developed by the W3C and WHATWG. It is still under standardization process and is believed to be fully established in 2014. HTML5 has new functions and/or attributes to improve the expressiveness of web[14].

### A. HTML5 Specific Attributes

In HTML5, various elements, attributes, APIs have been added to the previous version. Some of the security concerns have already been reported [15]. Amount of the elements added newly, as typical ones are as the following.

- Elements for defining the document structure ( such as `section`).
- Elements for defining the content structure (such as `article`, `main`, `aside`, etc.).
- Elements for defining the multimedia contents (`audio` and `video`).
- Elements for key pair generation (`keygen`).

Additionally, deprecated definitions and modifications of definitions are defined.

In addition, new APIs related to these additional elements are defined. e.g. Multimedia API, micro data API [16], 2d context of canvas element [17], web messaging API [18], background execution [19], two-way communication between client and server [20], data pushing from server to client [21].

Especially in the webStorage APIs for client-side storage (localStorage, sessionStorage) [22] are the key feature to implement the application can be comparable with native application. Furthermore, we are paying attention to these APIs because the Degree of freedom on web client can be magnified with the help of these APIs. We are expecting the evidences left by the web client will be more effective for investigation.

### B. WebStorage

WebStorage is one of the features added in HTML5. It is a mechanism for recording the data on the client side entity of web network (web browser), similar to cookies. Cookie is well known mechanism to provide the storage within web browser. Furthermore, it is about the add-on feature, flash cookie is used to store data on client side. Soltani [23] and Ayenson [24] survey usage of flash cookies for popular Web sites.

Although, webStorage has cleared away some limitations of cookie, for example, the capacity of webStorage is substantially larger than one of cookie. Storage spaces of webStorage are managed by *origin* [25]. *Origin* is the attribute of web server. Storage spaces are isolated by the unit defined by its *origin* and capacity of its unit is specified. Its capacity is different depending on the web browsers but never fall below 5 megabytes.

This capacity is significantly enlarged as compared to that of cookie. Differences between webStorage and cookie are storage capacity and the programming interface. WebStorage is only visible from client-side script. WebStorage data are never sent directly to server-side via HTTP without the client-side script processing.

### C. WebStorage APIs

WebStorage APIs are designed as the reflection of *key - value* store structure. The structure of webStorage consists of the tuples of *key* and *value*. Access methods for webStorage is defined as below.

`setItem(key, value)`
: Check the presence of entry with the `key` supplied as an argument. If it is absent, `key`/`value` pair supplies as arguments are added to the list as value the value corresponding to the `key`. If there is an entry with the `key` supplied as argument, its entry is updated with the `value` supplied as an argument. If cannot add/modify the entry, it raises exception.

`getItem(key)`
: Return the `value` of the entry that has the `key` supplied as an argument. If the entry corresponding to the `key` supplied as an argument does not exist, return `null`.

`removeItem(key)`
: Delete the entry that has the `key` supplied as an argument. If the entry corresponding to the `key` supplied as an argument does not exist, do nothing.

`clear()`
: Delete all entries. If there is no entries already, do nothing.

`key(n)`
: Return the `key` for *n*-th in the list of entries. The order of entries in the list is preserved as long as there is no addition or deletion of an entry. If argument is larger than the number of entries, return null.

`length`
: Return the number of entries.

There are two classes of webStorage, localStorage and sessionStorage. The difference between both is in the period of data is preserved.

*1) SessionStorage:* SessionStorage retains the data during the period of execution of the script which can access the data, except following reasons.

- Explicit remove method call.
- Limitation of storage space.
- Security reasons.

When top level browsing context is destroyed (i.e. when it becomes impossible to access the data), data is discarded.

When `Document` object is created in the browsing context that has the top level browsing context, browser confirms the existence of sessionStorage area corresponding to the origin of the `Document`. If there is no area, new sessionStorage areas are created.

*2) LocalStorage:* When the localStorage attribute is accessed, browser processes the following procedure.

- If the request may cause the policy violation (e.g. browser is not permitted to record persistent data), raise SecurityError exception.
- Confirm the format of *origin*. If its format does not consist of scheme, host and port, raise SecurityError exception and abort the procedure.
- Confirm whether the localStorage area correspond to *origin* of `Document` is allocated or not. If there is no allocated area, create new storage area correspond to the *origin*.
- Return the Storage object correspond to the localStorage area of origin.

*3) Origin: Origin* is a unit of persistent area of contents. It is introduced in order to ensure confidentiality and completeness of contents. In webStorage, this concept is leveraged for isolation and capacity management of storage area. *Origin* consists of:

- Protocol scheme
- Fully qualified host name
- Port number

If this combination matches, it will be considered as same *origin*. WebStorage area is isolated in units of *origin*.

## IV. Preliminary Experiments

### A. Objective of Experiment

In the preliminary experiments, we examine the possibility to retrieve evidence of web browsing activity. We confirm whether it is possible to retrieve the contents written in webStorage from the main memory image. First, open the test page and in the script page write out the message to webStorage. Then exit the web browser and acquire the main memory image and analyze.

### B. Experiment Environment

Experiment environments in the preliminary experiment are described as below. We used the AccessData FTKimager [26] as the most important memory imaging tool. Cory [27] describe a forensics that utilize open source tools. As a preliminary experiment, web browsers are run on virtual machine.

Host environment
: Microsoft Windows 7 Home Premium (64bit)
Main memory: 8 GB

Virtual Machine
: Oracle VirtualBox 4.3.2

Guest OS
: Microsoft Windows 7 Ultimate (64bit)
CPU: Allocated single core
Main memory: allocated 4GB

Web browser
: Mozilla Firefox 25.0.1

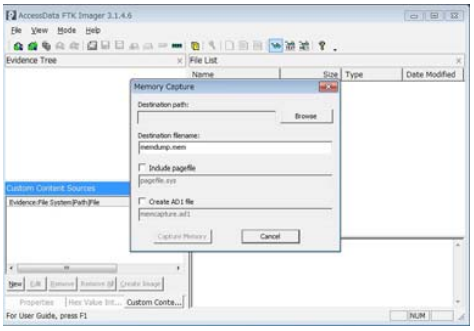


Fig. 1. FTK Imager

Google Chrome 31.0.1650.63m
Microsoft Internet Explorer 10.9.9200.16750
Memory image acquisition tool
AccessData FTK Imager 3.1.4.6 (Figure. 1)

### C. Experiment System Schematics

Schematics of the preliminary experiment system are shown in Figure 2. Targeted web browser is executed on the guest operating system runs on hypervisor. Furthermore, FTK Imager that acquires the main memory image runs on the guest operating system. Web server operates the remote machine through the network. The virtual bridge runs in the hypervisor connects internal and external networks. Script code within web page used for the experiments is described in Appendix.

### D. Experiment Procedure

First, the web page is loaded from remote web server. JavaScript code contained in the web page is written the message contained in the page to webStorage. At this point, it is necessary to prevent the message to appear explicitly on the network or web page. For this reason, the message is converted using rot13 (Figure 3). JavaScript code reads the message, converts it using rot13 and writes it to the webStorage. In the each experiment, after starting the guest Virtual Machine on the client terminal and launching the web browser, load the web page on the server.

After loading the page, write a message to webStorage is performed. After confirming the displayed page, exit web browser. Then acquire the whole of main memory image within the guest VM using FTK Imager. Finally, shutdown the guest operating system. Repeat the steps using different browsers.

### E. Experiment Result

*1) Mozilla Firefox:* Succeeded in retrieving the stored contents in the localStorage from the memory image obtained after running the Firefox. Retrieved contents contain the set of strings that denote *origin*, *key* and *value*. These sets appear two times in the whole image.

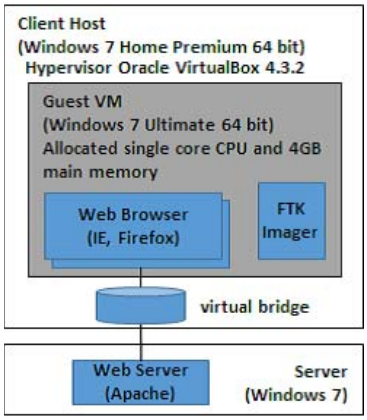


Fig. 2. Preliminary Experiment System

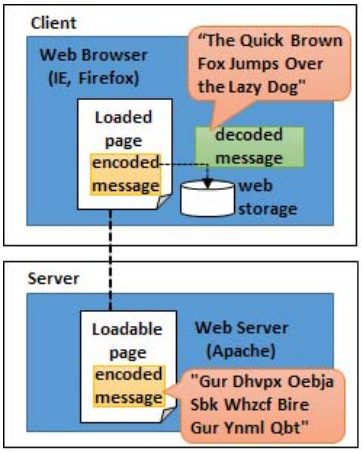


Fig. 3. Preliminary Experiment Schematics

*2) Microsoft Internet Explorer:* Then, we examined on Internet Explorer. Succeeded in retrieving the stored contents in localStorage from memory image obtained after running the Internet Explorer. Retrieved contents contains the set of strings that denote *origin*, *key* and *value*. Results shows *key* and string denoting *value* is encoded as DOM format. Furthermore, these are contained in the neighborhood of original message. Additionally, time attributes are encoded within the element(`ltime` and `htime`)

### F. Discussions on Preliminary Experiments

From the results of preliminary experiments, these are unveiled:

- memory residue
- format of evidence

## V. Experiments on Physical Environment

As preliminary experiments, we use virtual environment. However, it is not suitable to examine the memory forensics. Memory management mechanism equipped by hypervisor may intervene between host machine's memory and guest machine's. To investigate the memory image strictly, experiment must be physical environment.

### A. Experiment Environment

Experiment environments in these experiments are described below. Schematics of experiment system is shown in Figure 3. We choose 3 web browsers. Mozilla Firefox, Google Chrome and Microsoft Internet Explorer. These are the top three for Windows web browsers market share.

Client Machine
OS: Microsoft Windows 7 Ultimate (64bit)
CPU: AMD Phenom II X4 905e processor
Main memory: 4GB (Allocated 512MB to Video)
Web browser
Mozilla Firefox 25.0.1
Google Chrome 31.0.1650.63m
Microsoft Internet Explorer 10.9.9200.16750

Memory image acquisition tool

AccessData FTK Imager 3.1.4.6 [26]

Memory wipe

Memtest86+ 5.01

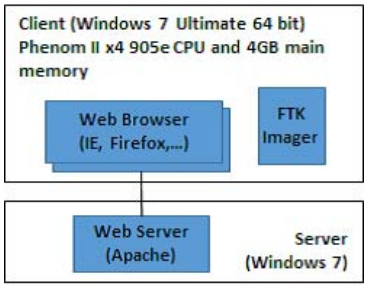


Fig. 4. Experiment System

### B. Experiment System Schematics

Experiments conducted this section is almost the same as preliminary experiments described in IV-C. However, preliminary experiments are conduced on virtual environment. We should notice that

### C. Experiment Procedure

In this experiments, procedure is as following.

1) Boot the client machine with Memtest86+ and execute memory test(at least 1 pass).
2) Shut the client machine down and reboot with Windows OS.
3) Launch FTK Imager.
4) Launch target web browser.
5) Load the storage test page (local/session storage) on the web browser.
6) Exit the web browser.
7) Acquire memory image with FTK Imager.
8) Connect external hard disk drive and move image file.
9) Shut the client machine down.

Especially, In the experiment procedure, memory test in first step characterize the experiments in this section. This test definitely overwrite the entire main memory and wipes the residual value.

### D. Experiment Result

*1) Mozilla Firefox:*

*a) LocalStorage:* In the first place, we examined on localStorage of Firefox. In this experiment, we succeed in retrieving contents and confirmed the result of the experiment in previous section that succeeded to retrieve the evidence of localStorage in virtual environment.

Furthermore, succeeded to retrieve the contents of sessionStorage.

Excerpt of hexadecimal dump of memory image retrieved in this experiment is shown below.

```
012a1ffa0|00 00 00 00 00 00 00 00 00 00 54 01 06 35 2b 63
012a1ffb0|00 00 36 2e 30 2e 38 36 31 2e 32 39 31 2e 3a 68
012a1ffc0|74 74 70 3a 38 30 6c 6f 63 61 6c 53 74 6f 72 61
012a1ffd0|67 65 4b 65 79 54 68 65 20 51 75 69 63 6b 20 42
012a1ffe0|72 6f 77 6e 20 46 6f 78 20 4a 75 6d 70 73 20 4f
012a1fff0|76 65 72 20 54 68 65 20 4c 61 7a 79 20 44 6f 67
```

For readability, character dumped result image is shown below.

```
012a1ffa0|..........T..5+c..6.0.861.291.:h
012a1ffc0|ttp:80localStorageKeyThe Quick B
012a1ffe0|rown Fox Jumps Over The Lazy Dog
```

As found in the first glance, *origin* and string denoting *key* and *value* is contained with concatenated form. Furthermore, the address part of the *origins* contained in the reverse order. From this evidence, investigator can search the location of required data based on the knowledge of *origin*.

*b) SessionStorage:* Next, we examined on the sessionStorage of Firefox. Excerpt of hexadecimal dump of memory image is show below.

```
10cb411b0|2e 30 2e 36 2f 66 61 76 69 63 6f 6e 2e 69 63 6f
10cb411c0|22 2c 22 69 6e 64 65 78 22 3a 32 2c 22 73 74 6f
10cb411d0|72 61 67 65 22 3a 7b 22 68 74 74 70 3a 2f 2f 31
10cb411e0|39 32 2e 31 36 38 2e 30 2e 36 22 3a 7b 22 73 65
10cb411f0|73 73 69 6f 6e 53 74 6f 72 61 67 65 4b 65 79 22
10cb41200|3a 22 54 68 65 20 51 75 69 63 6b 20 42 72 6f 77
10cb41210|6e 20 46 6f 78 20 4a 75 6d 70 73 20 4f 76 65 72
10cb41220|20 54 68 65 20 4c 61 7a 79 20 44 6f 67 22 7d 7d
10cb41230|7d 5d 2c 22 73 65 6c 65 63 74 65 64 22 3a 31 2c
10cb41240|22 5f 63 6c 6f 73 65 64 54 61 62 73 22 3a 5b 5d
```

Character dumped image is shown below.

```
10cb411b0|.0.6/favicon.ico","index":2,"sto
10cb411d0|rage":{"http://192.168.0.6":{"se
10cb411f0|ssionStorageKey":"The Quick Brow
10cb41210|n Fox Jumps Over The Lazy Dog"}}
10cb41230|}],"selected":1,"_closedTabs":[]
```

It is clear that *origin*, *key* and *value* are encoded as JSON format. View as a tree, format is in the form of *key* and *value* are hanging under the *origin*.

*2) Microsoft Internet Explorer:* Next, we examined on Microsoft Internet Explorer. Similar to preliminary experiment, succeeded to retrieve the contents of localStorage. Furthermore, succeeded to retrieve the contents of sessionStorage.

*a) LocalStorage:* About Internet Explorer, succeed to retrieve the stored contents in localStorage from memory image obtained after running browser.

Excerpt of hexadecimal dump of memory image is show below.

```
03f8a55a0|83 00 00 00 18 00 00 00 3c 72 6f 6f 74 3e 3c 69
03f8a55b0|74 65 6d 20 6e 61 6d 65 3d 22 6c 6f 63 61 6c 53
03f8a55c0|74 6f 72 61 67 65 4b 65 79 22 20 76 61 6c 75 65
03f8a55d0|3d 22 54 68 65 20 51 75 69 63 6b 20 42 72 6f 77
03f8a55e0|6e 20 46 6f 78 20 4a 75 6d 70 73 20 4f 76 65 72
03f8a55f0|20 54 68 65 20 4c 61 7a 79 20 44 6f 67 22 20 6c
03f8a5600|74 69 6d 65 3d 22 36 32 30 31 34 33 34 37 32 22
03f8a5610|20 68 74 69 6d 65 3d 22 33 30 33 35 35 35 39 33
03f8a5620|22 20 2f 3e 3c 2f 72 6f 6f 74 3e 00 00 00 00 00
03f8a5630|ff ff ff ff 82 79 47 11 00 00 00 00 00 00 00 00
```

Character dumped image is show below.

```
03f8a55a0|........<root><item name="localS
03f8a55c0|torageKey" value="The Quick Brow
03f8a55e0|n Fox Jumps Over The Lazy Dog" l
03f8a5600|time="620143472" htime="30355593
03f8a5620|" /></root>...........G.........
```

It is clear that *key* and *value* are contained as DOM format in the memory. It is confirmed the result of preliminary

TABLE II
EXPERIMENTAL RESULT

| Browser | storage type | Format | origin | key | value |
|---|---|---|---|---|---|
| Mozilla Firefox | localStorage | concatenated string | plain, but address part is inverted | plain format | plain format |
| | sessionStorage | JSON format | plain format | plain format | plain format |
| MS Internet Explorer | localStorage | DOM format | not observed | plain format | plain format |
| | sessionStorage | concatenated string | not observed | plain format | UTF-16 |
| Google Chrome | localStorage | concatenated string | not observed | UTF-16 | UTF-16 |
| | sessionStorage | concatenated string | not observed | plain format | UTF-16 |

experiment. There are time attributes are encoded within the element.

*b) SessionStorage:* Next, we examined on sessionStorage. Excerpt of hexadecimal dump of memory image is shown below.

```
06c5a2d30|00 06 08 01 01 15 00 00 00 00 00 00 31 06 19 00
06c5a2d40|73 65 73 73 69 6f 6e 53 74 6f 72 61 67 65 4b 65
06c5a2d50|79 00 17 00 00 00 00 00 00 17 08 56 01 16 00 00
06c5a2d60|00 00 00 00 54 00 68 00 65 00 20 00 51 00 75 00
06c5a2d70|69 00 63 00 6b 00 20 00 42 00 72 00 6f 00 77 00
06c5a2d80|6e 00 20 00 46 00 6f 00 78 00 20 00 4a 00 75 00
06c5a2d90|6d 00 70 00 73 00 20 00 4f 00 76 00 65 00 72 00
06c5a2da0|20 00 54 00 68 00 65 00 20 00 4c 00 61 00 7a 00
06c5a2db0|79 00 20 00 44 00 6f 00 67 00 00 37 00 6e 61 6d
06c5a2dc0|65 73 70 61 63 65 2d 44 43 33 41 46 35 44 45 5f
06c5a2dd0|42 46 37 33 5f 34 35 38 41 5f 41 46 45 42 5f 38
06c5a2de0|45 44 32 46 32 37 32 44 31 38 41 2d 00 1b 00 00
```

Character dumped image is shown below.

```
06c5a2d30|............1...sessionStorageKe
06c5a2d50|y.........V........T.h.e. .Q.u.
06c5a2d70|i.c.k. .B.r.o.w.n. .F.o.x. .J.u.
06c5a2d90|m.p.s. .O.v.e.r. .T.h.e. .L.a.z.
06c5a2db0|y. .D.o.g..7.namespace-DC3AF5DE_
06c5a2dd0|BF73_458A_AFEB_8ED2F272D18A-....
```

Appearance of this result is quite different from the result of localStorage. *value* is encoded as UTF-16. Furthermore, there are some noise in memory, and string "`namespace`" can be observed.

*3) Google Chrome:* Finally, we examined on Google Chrome browser.

*a) LocalStorage:* We succeed to retrieve the contents of localStorage from main memory image obtained after running Google Chrome.

Excerpt of hexadecimal dump of memory image is shown below.

```
0370cc710|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0370cc720|78 03 04 49 81 38 6c 00 6f 00 63 00 61 00 6c 00
0370cc730|53 00 74 00 6f 00 72 00 61 00 67 00 65 00 4b 00
0370cc740|65 00 79 00 54 00 68 00 65 00 20 00 51 00 75 00
0370cc750|69 00 63 00 6b 00 20 00 42 00 72 00 6f 00 77 00
0370cc760|6e 00 20 00 46 00 6f 00 78 00 20 00 4a 00 75 00
0370cc770|6d 00 70 00 73 00 20 00 4f 00 76 00 65 00 72 00
0370cc780|20 00 54 00 68 00 65 00 20 00 4c 00 61 00 7a 00
0370cc790|79 00 20 00 44 00 6f 00 67 00 64 01 04 21 81 38
0370cc7a0|6c 00 73 00 4b 00 65 00 79 00 54 00 68 00 65 00
0370cc7b0|20 00 51 00 75 00 69 00 63 00 6b 00 20 00 42 00
0370cc7c0|72 00 6f 00 77 00 6e 00 20 00 46 00 6f 00 78 00
0370cc7d0|20 00 4a 00 75 00 6d 00 70 00 73 00 20 00 4f 00
0370cc7e0|76 00 65 00 72 00 20 00 54 00 68 00 65 00 20 00
0370cc7f0|4c 00 61 00 7a 00 79 00 20 00 44 00 6f 00 67 00
0370cc800|0a 00 00 00 02 03 ce 00 03 ce 03 f1 00 00 00 00
```

Character dumped image is shown below.

```
0370cc710|................x..I..l.o.c.a.l.
0370cc730|S.t.o.r.a.g.e.K.e.y.T.h.e. .Q.u.
0370cc750|i.c.k. .B.r.o.w.n. .F.o.x. .J.u.
0370cc770|m.p.s. .O.v.e.r. .T.h.e. .L.a.z.
0370cc790|y. .D.o.g.d..!..l.s.K.e.y.T.h.e.
0370cc7b0| .Q.u.i.c.k. .B.r.o.w.n. .F.o.x.
0370cc7d0| .J.u.m.p.s. .O.v.e.r. .T.h.e. .
0370cc7f0|L.a.z.y. .D.o.g.................
```

Similar to sessionStorage of Internet Explorer, *value* is encoded as UTF-16. In centrally to Internet Explorer, *key* is encoded as UTF-16, too. *Origin* is not observed in memory image.

*b) SessionStorage:* Excerpt of hexadecimal dump of memory image is shown below.

Succeed to retrieve the contents of sessionStorage from main memory image obtained after running Google Chrome.

```
0367940a0|31 2d 01 31 01 17 6d 61 70 2d 31 2d 73 65 73 73
0367940b0|69 6f 6e 53 74 6f 72 61 67 65 4b 65 79 56 54 00
0367940c0|68 00 65 00 20 00 51 00 75 00 69 00 63 00 6b 00
0367940d0|20 00 42 00 72 00 6f 00 77 00 6e 00 20 00 46 00
0367940e0|6f 00 78 00 20 00 4a 00 75 00 6d 00 70 00 73 00
0367940f0|20 00 4f 00 76 00 65 00 72 00 20 00 54 00 68 00
036794100|65 00 20 00 4c 00 61 00 7a 00 79 00 20 00 44 00
036794110|6f 00 67 00 77 d2 97 03 d3 00 01 17 00 00 00 00
```

Character dumped image is shown below.

```
0367940a0|1-.1..map-1-sessionStorageKeyVT.
0367940c0|h.e. .Q.u.i.c.k. .B.r.o.w.n. .F.
0367940e0|o.x. .J.u.m.p.s. .O.v.e.r. .T.h.
036794100|e. .L.a.z.y. .D.o.g.w..........
```

*Value* is encoded as UTF-16 and *key* is contained same to original.

## VI. DISCUSSIONS

Summary of the experimental result in the previous section is as shown as Table II.

As a result of experiments on all of the three browsers, contents of webStorage have been retrieved from main memory image. It is similar for three types of web browsers. Especially for Mozilla Firefox, residual of *origin* can be retrieved from the neighboring evidence contents.

## VII. CONCLUSION AND FUTURE WORK

In the experiments, we succeeded in retrieving the webStorage contents of Firefox, Internet Explorer and Chrome from main memory. Although the formats differ, it succeeded for all three browsers in both the local and sessionStorage.

While investigating the evidence in digital forensics, it may be necessary to handle the large image file. Its size may reach

several gigabytes in general. Therefore, whether or not the search can be performed machinery is important. From this point of view, it is important whether or not it is possible to search the information of *origin* when *key* is not known. About Mozilla Firefox, *origin* is contained in the neighbor of contents.

In the matter of Microsoft Internet Explorer and Google Chrome, evidence for the *origin* is not observed neighbor of contents. Therefore, it is difficult to track down the content when only *origin* has been identified. As long as the information about the *key* is known, it is possible to find out the location of *value* in the memory image. Furthermore, it is clear that the formats of contents of webStorage reflect the internal data structure of the browsers.

Further experiments and studies are necessary in the mechanism of web browser whether or not the processing data remains in the memory. It is also necessary to perform the analysis on the source code for Mozilla Firefox and Google Chrome. As for Internet Explorer, it is necessary to seek a different approaches to perform the steps.

## REFERENCES


[1] J. Sammons, *The Basics of Digital Forensics*. Elsevier, 2012.

[2] M. Tito, "Forensic analysis of the firefox 3 internet history and recovery of deleted sqlite records," *Digital Investigation: The International Journal of Digital Forensics & Incident Response archive*, vol. 5, pp. 93–103, March 2009.

[3] J. Oh, S. Lee, and S. Lee, "Advanced evidence collection and analysis of web browser activity," *Digital Investigation: The International Journal of Digital Forensics & Incident Response archive*, vol. 8, pp. S62–S70, August 2011.

[4] D. J. Ohana and N. Shashidhar, "Do private and portable web browsers leave incriminating evidence? a forensic analysis of residual artifacts from private and portable web browsing sessions," *Security and Privacy Workshop (SPW)*, pp. 135–142, May 2013.

[5] M. Mulazzani, "New challenges in digital forensics: online storage and anonymous communication," Ph.D. dissertation, Vienna University of Technology, 2014.

[6] K. Satvat, M. Forshaw, F. Hao, and E. Toreini, "On the privacy of private browsing - a forensic approach," in *Data Privacy Management and Autonomous Spontaneous Security*. Springer Berlin Heidelberg, 2014, pp. 380–389.

[7] S. P. Aditya Mahendrakar, James Irving, *Forensic analysis of private browsing artifacts*. IEEE, 2011, pp. 197–202.

[8] G. Aggarwal, E. Bursztein, C. Jackson, and D. Boneh, "An analysis of private browsing modes in modern browsers." in *USENIX Security Symposium*, 2010, pp. 79–94.

[9] W. Jansen and R. Ayers, "Guidelines on cell phone forensics," *NIST Special Publication*, vol. 800, p. 101, 2007.

[10] S. Willassen, "Forensic analysis of mobile phone internal memory," in *Advances in Digital Forensics*. Springer, 2005, pp. 191–204.

[11] R. Ahmed and R. V. Dharaskar, "Mobile forensics: an overview, tools, future trends and challenges from law enforcement perspective," in *6th International Conference on E-Governance, ICEG, Emerging Technologies in E-Government, M-Government*, 2008, pp. 312–23.

[12] K. Amari. (2009) Techniques and tools for recovering and analyzing data from volatile memory. SANS Institute. [Online]. Available: http://www.sans.org/reading-room/whitepapers/forensics/ techniques-tools-recovering-analyzing-data- volatile-memory-33049

[13] A. Waksman and S. Sethumadhavan, "Silencing hardware backdoors," *SP '11 Proceedings of the 2011 IEEE Symposium on Security and Privacy*, pp. 49–63, 2011.

[14] S. Pieters. (2013) Differences from html4. [Online]. Available: http://www.w3.org/TR/2013/WD-html5-diff-20130528/

[15] European Network and Information Security Agency. (2011) A security analysis of next generation web standards. [Online]. Available: https://www.enisa.europa.eu/activities/Resilience-and-CIIP/critical-applications/web-security/a-security-analysis-of-next-generation-web-standards

[16] I. Hickson. (2013) Html microdata. [Online]. Available: http://www.w3.org/TR/2013/NOTE-microdata-20131029/

[17] R. Cabanier, E. Graff, J. Munro, T. Wiltzius, and I. Hickson. (2013) Html canvas 2d context. [Online]. Available: http://www.w3.org/TR/2013/CR-2dcontext-20130806/

[18] I. Hickson. (2012) Html5 web messaging. [Online]. Available: http://www.w3.org/TR/webmessaging/

[19] ——. (2012) Web workers. [Online]. Available: http://www.w3.org/TR/2012/CR-workers-20120501/

[20] ——. (2012) The websocket api. [Online]. Available: http://www.w3.org/TR/2012/CR-websockets-20120920/

[21] ——. (2012) Server-sent events. [Online]. Available: http://www.w3.org/TR/2012/CR-eventsource-20121211/

[22] ——. (2013, July) Web storage w3c recommendation 30 july 2013. [Online] http://www.w3.org/TR/2013/REC-webstorage-20130730/.

[23] A. Soltani, S. Canty, Q. Mayo, L. Thomas, and C. J. Hoofnagle. (2009) Flash cookies and privacy. [Online]. Available: http://ssrn.com/abstract=1446862

[24] M. D. Ayenson, D. J. Wambach, A. Soltani, N. Good, and C. J. Hoofnagle. (2011) Flash cookies and privacy ii:now with html5 and etag respawning. [Online]. Available: http://ssrn.com/abstract=1898390

[25] A.Barch. (2011) The web origin concept. RFC6454.

[26] Computer forensics software for digital investigations. AccessData. [Online]. Available: http://www.accessdata.com/products/digital-forensics/ftk

[27] C. Altheide and H. Carvey, *Digital Forensics with Open Source Tools*. Syngress, 2011.


## APPENDIX

Script code written with JavaScript is indicated as following. This script part simply put the entry to webStorage. Entry is consisted of *key* and *value*, but *value* is converted with rot13 algorithm prior to put to webStorage. Function `rot13` encodes /decodes the message supplied as argument.

This script is for experiments on localStorage. For sessionStorage, *key* value and type of webStorage is altered to for sessionStorage.

```
<script type="text/javascript">

  var key = "localStorageKey";
  var dispArea = document.getElementById("dispArea");
  origText = "Gur Dhvpx Oebja Sbk Whzcf Bire Gur Ynml Qbt"
  localStorage.setItem(key, rot13(origText));

  if(origText != null){
    dispArea.value = origText;
  }

  function onClear(){
    localStorage.setItem(key,"");
    dispArea.value = "";
  }
    ...
</script>
```