Study on Attack Scenarios with HTML5

Soojin Yoon, Jonghun Jung, HwanKuk Kim
Mobile Security R&D Team
KISA (Korea Internet & Security Agency)
Seoul, Korea

Abstract

The new Web standard HTML5 makes a Web page provide dynamic functions to users without additional plug-ins such as ActiveX, Flash and Silverlight. Most attacks on Web browsers use such plug-ins. HTML5 provides the abilities that can be substituted to plug-ins, so hackers focus their attacks using HTML5. This paper surveys 16 attacks using HTML5 presented and shows their effects. Additionally, three new attacks using HTML5 are also discussed

1. Introduction

HTML is a computer language for Web pages. Most Web pages use HTML in writing or presentations. As Web use increases, the use of HTML has grown in tandem. Users want to use several functions on a Web browser, but the functionality of HTML has been sufficient to satisfy users. After HTML4 was announced in 1997, no additional improvement in HTML came before HTML5. Plug-ins like ActiveX and Flash appeared to allow Web pages to run dynamic functions, but side effects arose. Attackers are fond of the plug-ins when they attack, making Web pages messy with these plug-ins. Recently, the newly announced standard HTML5 runs new APIs. Web users can use HTML5 instead of plug-ins when writing their Web pages.

HTML5 replaces plug-ins but comes with side effects as well. Attackers consider HTML5 a new attack tool and method. In particular, a DoS attack with HTML5 is effective. On April 2, 2014, the world's eighth-largest website SOHU TV came under a DDoS attack through an HTML5 function [1].

An attack using HTML5 takes on user browsers but leaves no trace like an executable file or specific packet. Though attacks using HTML5 are rare, their effects are unpredictable.

This paper describes 19 attacks using HTML5 -- 16 of which were announced by other papers or demos and three new. It shows how HTML5 can be used for attack and the possibility of an attack and effects.

2. HTML5

HTML is a programming language for writing Web pages and the newest version of HTML. It was processed from 2009 and confirmed as a new Web standard on October 28, 2014.

The major features of HTML5 are semantic tags, CSS3 and new APIs.

Semantic tags offer information on Web content through tags. Before the tags existed, Web programmers used to write tag to show content type. For example, they wrote to show a picture on a Web page between div tags. In HTML5, a semantic tag <figure> marks pictures or graphic content. Moreover, semantic tags like <menu>, <nav> are used for writing semantic information using tags.

CSS3 (Cascading Style Sheet 3) is a language providing the visual structure of Web pages. Its dynamic graphic support includes functions for media and animation, but it could not implement in CSS2.

New APIs are written in Javascript, an object-based language offering dynamic functions on a Web page. Certain APIs are recommended while others remain in progress. Famous new APIs are used in Web browsers such as WebWorker, WebSocket and Indexed DB. Details of new APIs are explained in this paper's description of an attack scenario using them.

3. Preparation of ATTACK

3.1. Outline of HTML5 Attack

Attackers make a malicious Web page like a phishing site or inject malicious Javascript into a regular Web page. Users then perform malicious behavior on their browsers when accessing the Web page the attacker inserted malicious Javascript into.

An attack using HTML5 runs on a user's browser. When the target traces the source of the attack, the user accessing the malicious Web page is the starting point.

3.2. Javascript Injection

Malicious Javascript injection is known as XSS (Cross Site Scripting). Attackers inject Javascript into a Web page to exploit its vulnerability. Many methods of injection exist, but this paper focuses on cases using HTML5 features.

3.2.1. Using new tags of HTML5. Javascript injection can be with new HTML5 tags like <video>, <audio>.

<video poster=javascript:alert(1)//></video>

The upper line is an example of Javascript's injection into a poster attribute of video tag. A poster attribute shows an image before playing a video clip.

Running Javascript injected with a new tag depends on the Web browser because the support tags from each browser are different.

Other methods of Javascript injection are omitted because of a paper shortage.

3.2.2. Using CSS3. CSS3 has various elements for providing visual expression, and among them, -o-link and -o-link-source indicate links. Attackers use these elements for injecting Javascript.

X

The example of style tag runs Javascript though the –o-link-source element. It also depends on which Web browser opens a Web page.

3.2.3. Using SVG. SVG is an image format expressing an image as vectors. HTML5 provides the <svg> tag to implement an svg image on a Web page. Tags in svg are defined to express the image. SVG is also saved as a file.

SVG file and <svg> tag can include <script> tag. When a Web browser loads an SVG file or <svg> tag, <script> tag included in svg is loaded and executed as well [2], [3].

3.2.4. Injection into HTML5-based Apps. HTML5-based apps are mobile or Web apps developed by HTML5 and Javascript. If there is an input channel like QR code scanner, this is the vulnerability of Javascript injection [4]. When some value from outside apps (missing verb???), most HTML5-based apps don't care whether it is Javascript or valid value. In the case of Javascript, the injected command is automatically active. The known weak channels are wifi ssd, file name, media meta data, QR code, URL and etc.

3.2.5. Using Canvas. Canvas is a new element of HTML5 supporting 2-D or 3-D graphics. It is an injection of iframe, but can be misused for injection of Javascript. Canvas loads a malicious PNG file which meta data is code creating an iframe. As a PNG file is loading, the attacker's iframe is created and its code get active.[5]

4. Attacks using HTML5

Attacks using HTML5 abuse new APIs and semantic tags. The attacks below are classified by damage type.

Table 1. Attacks with HTML5

Type	Attack
Data Leaks	Click Jacking
	Leak of WebStorage
	WebSocket Hijacking
	Leak of WebSocket Data
	Keylogger by SVG
	Mouse Logger
	Network Scan
	Leak of Geolocation Information
Information Manipulation	WebStorage Manipulation
	Indexed DB Manipulation
	WebCache Manipulation
	History Manipulation
Request Forgery	CSRF with CORS
	WebSocket Hijacking
	CrossPrinting
DoS Attack	WebStorage Fill Up
	WebWorker Botnet
	ClientDoS
	Server-sent Event Botnet
	Vibration Attack
Social Engineering	**Phishing Notification**
	Autocomplete Input Form Leak
	Fake Telephone Call

4.1. Data Leak

As HTML5 provides a variety of functions, the types of information vary. Moreover, connecting APIs such as WebSocket and XMLHttpRequest are added to HTML5. This makes the information of a Web page sent to other Web pages or users.

Data Leaks is a type of attack in which user information is sent to attackers without user awareness.

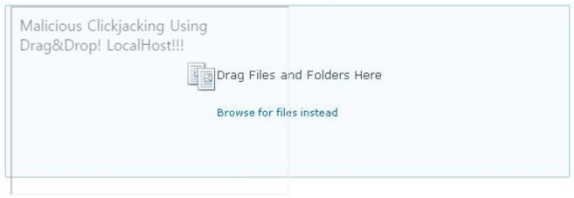

Figure 1. ClickJacking example
(An invisible iframe locates on the left side of the original area for file upload)

4.1.1. Clickjacking ClickJacking deceives users through an invisible iframe. In CSS3, the opacity of iframe can be adjusted. An attacker places an invisible iframe on an original area. When users click or drop files in the area, the attacker's invisible iframe intercepts a user click or file.[6]

When attackers try to steal a user file, they use Drag-Drop attribute. It gets a file on an area users drag and drops the file inside. If the file is dropped on an attacker's area, the attacker's iframe can send it to attackers with connecting APIs like XMLHttpRequest [3], [7]. Fig. 1 is an example of an attacker attempting to steal a user file with Drag-Drop attribute. It shows the line and content of invisible iframe for explaining ClickJacking.

4.1.2. Leak of WebStorage Leak of WebStorage seeks to steal information from WebStorage [6]. WebStorage API saves or uses data as a pair of key values in each Web domain. WebStorage is the space that piles data with WebStorage API and composed of both LocalStorage and SessionStorage. Data of SessionStorage expires with the end of a session, while that of LocalStorage does not. Leak of WebStorage is on LocalStorage and picks user data from LocalStorage and sends it to the attacker with connecting APIs. If something valuable is in LocalStorage like passwords, the attack has major effects. Many technical documents recommend not putting sensitive data in WebStorage. If something personal needs saving, it should be encrypted.

4.1.3. WebSocket Hijacking WebSocket Hijacking is connecting an attacker's or target's WebSocket server without the user's permission [8], [9]. WebSocket API is a new API in HTML5, and provides functions for connection to a WebSocket server on a Web page. The connection of WebSocket operates like that of a socket. Though it can exchange data though WebSocket, notifying user connection or authenticating the user is not needed. It can connect any WebSocket server without user perception and be used to steal user information.

4.1.4. Leak of WebSocket Data Leak of WebSocket data is an attack that tries to steal data on a user connection though WebSocket [9]. When an attacker defines an on message handler in duplicate form, the latter activates and gets a message on WebSocket. If it sends the message to the attacker, he or she can get the message.

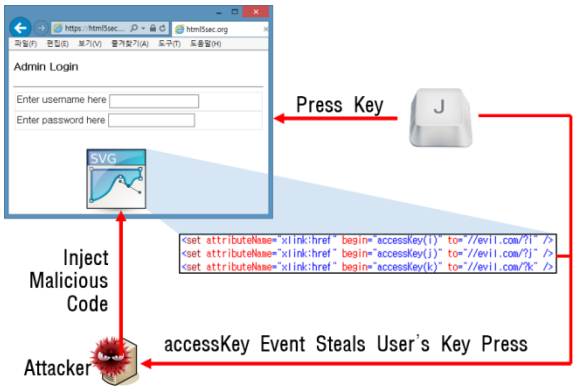

Figure 2. Keylogger by SVG

4.1.5. Keylogger by SVG As a keylogger gets the key user pressed, it can be implemented by accessKey event in SVG [10], [11], [12].

<set attributeName="xlink:href" begin="accessKey(j)" to="//evil.com/?j" />

The upper tag <set> is an SVG function occuring in a link access to 'evil.com/?j' when the user presses the key "J." If many tags <set> are in a Web page, the attacker can guess most of a user's key activities.

4.1.6. Mouse Logger A mouse logger gets a pointer where the user's mouse or touch is on. Before HTML5, the Javascript function gets a pointer coordinate of the mouse on a Web page. Mouse logger is an attack based on Javascript and one from an HTML5 attack either because of the pointer's API. HTML5 provides pointer API as standard, and the attacker uses the pointer API and gets mouse coordinate composed of a pointer API and connecting APIs.

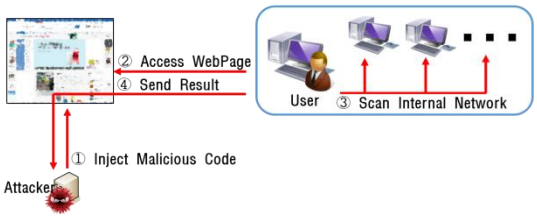

Figure 3. Network Scan

4.1.7. Network Scan Network scan gets information on a user's internal network using WebSocket, XMLHttpRequest [3], [13]. When a user accesses a Web page with network scan code, the user's browser sends network traffic to other machines belonging to a user's network. Then the browser sends the results to the attacker on which IP or port is

available. Network scan might be not critical, but information from a network scan can be used for another attack.

4.1.8. Leak of Geolocation Information Leak of geolocation information gets a user's geolocation with geolocation API in HTML5 [3]. If a user's device is portable, geolocation is sensitive information because user location can be guessed by the geolocation of device. When a geolocation API is used, the Web browser notifies the user to permit use of geolocation API. If a user is allowed to use geolocation API, the browser does not ask the user again. When the user accesses a website he or she has allowed to use geolocation API on, the browser runs geolocation API and cares not from which original code or injected malicious code it came from.

4.2. Information Manipulation

Information manipulation occurs when an attacker lets a user do an intended action by manipulating information on a user's Web browser.

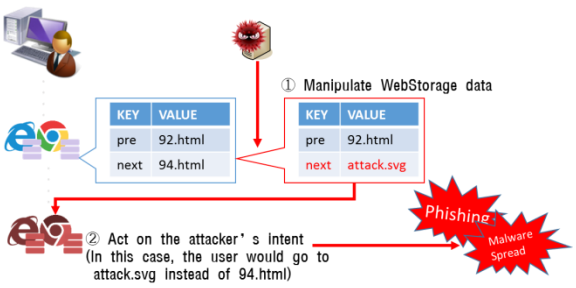

Figure 4. WebStorage Manipulation Example

4.2.1. WebStorage Manipulation. WebStorage manipulation occurs when an attacker changes data on LocalStorage [6]. The attacker manipulates data of LocalStorage by script injection and expects a user's browser to take action as he or she intended. Figure 4 shows an example occurring on an e-book site. The site uses WebStorage as a bookmark to where a user will or have read. It makes users go to an attacker's intended page.

4.2.2. IndexedDB Manipulation. IndexedDB was proposed and chosen as a recommendation. IndexedDB is a NoSQL DB and storage and API of treating values with distinct indexes. If no additional security steps are present, values are recorded as plain text. An attacker can change the values of IndexedDB, and attract users to do his or her intended action. Recently, several secure libraries for IndexedDB were developed by Web developers, not W3C.

4.2.3. WebCache Manipulation. WebCache stores the resource of a Web page and replaces a broken resource when offline. WebCache manipulation occurs when an attacker manipulates the resources of WebCache, places users offline and leads users to do malicious thing.[3][14] For this situation to occur is difficult, however. WebCache is also a theoretical attack, not a practical one.

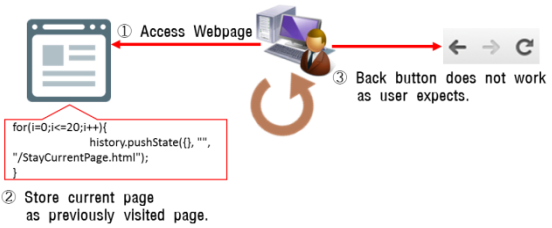

Figure 5. History Manipulation

4.2.4. History Manipulation History API controls the Web page history of a browser, and history manipulation misuses history API for attracting users to an attacker's page [3]. History API can add and delete a Web page's history. If too many pages are inserted into previous pages, a user cannot exit the current or inserted page by pressing the back button. In effect, this attack is trivial because history API can only treat the same domain page.

4.3. Request Forgery

Request forgery makes users perform malicious actions by sending a request to the attacker's target without user consent. This paper introduces two existing attacks and a new one we tested.

Figure 6. CSRF with CORS

4.3.1. CSRF with CORS. CSRF (Cross Site Request Forgery) with CORS (Cross Origin Resource Sharing) is an attack to send a deceptive request to other origins using CORS condition [3], [6]. CORS allows requests to be sent to other origins different from the domain of the current Web page. With HTML5, a browser can take action with CORS.

http://www.example.com/index.php?page=http://www.attacker.com/exploit.php

http://www.example.com/#http://www.attacker.com/stealDetails.php

The upper two URLs are examples that send requests to attacker.com [15]. To block CORS, a website should block requests from other origins.

Otherwise, the website is weak against CSRF with CORS.

In addition, the method content Window.postMessage shares resources between iframe and its parent page. If this method and inserted iframe also are used, the attacker can exhibit malicious behavior.

4.3.2. WebSocket Hijacking WebSocket hijacking is also classified as information manipulation. It is explained in the information manipulation section, so this paper omits the details in this section.

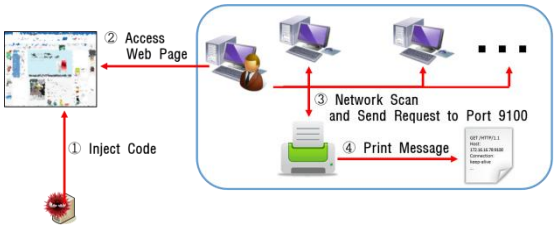

Figure 7. Cross Site Printing

4.3.3. Cross Site Printing. This paper introduces cross-site printing as a new attack form, however cross-site printing without HTML5 exist [16]. The previous cross-site printing sends a GET/POST request to a certain printer IP and lets it print the message. But cross-site printing with HTML5 does not care what IP belongs to the printer. As an attacker does network scan, he or she sends a request including the message of intent to print. If a printer gets this request and its configuration is default, the message of the request will be printed. This attack can be misused for spamming and bothering someone.

4.4. DoS Attack

DoS (Denial of Service) attack makes it difficult for users to use certain Web pages. Most cases involving sending a mass of messages to a target. This paper discusses three existing and two new attacks.

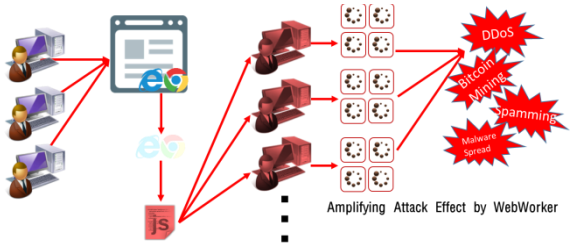

Figure 8. WebWorker Botnet

4.4.1. WebWorker Botnet. WebWorker creates another thread for running Javascript to divide workload. WebWorker is limited to access resource of Web page including itself for security but can send requests to other networks.

An attacker can consider users as bots with WebWorker [3], [15]. Users connect a malicious web page, then their browsers run WebWorker and its malicious Javascript. It can cause DDoS attacks, Bitcoin mining and spamming.

4.4.2. Client DoS. Client DoS bothers users to use a Web page [2]. Its target is a user's browser, not Web server. It is different from typical DoS in using a new attribute of HTML5 like a repeat pattern. It depends on the version and type of Web browser.

<input onblur=focus() autofocus>

Upper tag is one example of client DoS. This makes input form and it get keyboard focus when a Web page is loaded. It has the attribute onblur that defines action when it loses keyboard focus. And method focus () is a function to get keyboard focus. If injected into a Web page, a user connected to the Web page cannot move the keyboard focus outside it. It does not push load on the browser, but uses the Web page's disable.

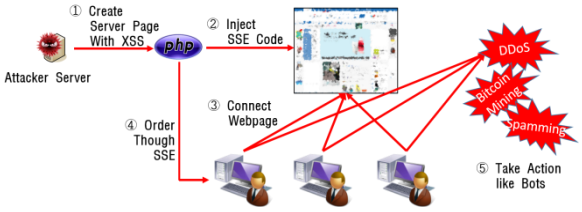

Figure 9. Server-Sent Event Botnet

4.4.3. Server-Sent Event Botnet. SSE (Server-sent event) replaces the pulling of ASP and PHP and gets the message of server with server-sent event.

An attacker can use SSE as channel for commanding bots. A Web page has the SSE of the attacker, then distributes commands to users connecting the Web page. Commands are written in Javascript or HTML. A user's browser runs them and takes malicious actions like DDoS, bitcoin mining or spamming.

The distinct feature here is that it is from ASP and PHP, not Javascript. The Javascript code of WebWorker can be easily read though obfuscated. But PHP or ASP of SSE shows no content and code. Analysis of SSE botnet takes a lot more time and effort than that of WebWorker botnet.

Figure 10. Vibration Attack

4.4.4. Vibration Attack. This is for mobile devices. HTML5 provides a vibration API for mobile devices, and makes such devices vibrate. Though it has limit on how long a device vibrates, an attacker can keep a device vibrating with the loop function.

We tested this attack, and wrote code composed of vibration API and setInterval of loop function and calls certain function with interval. Our phones accessed the test Web page, the code took effect and phones vibrated until we closed the test Web page.

4.5. Social Engineering

Social Engineering deceives users by their carelessness. It is not technical hacking, but psychological manipulation.

4.5.1. Phishing Notification. Desktop notification sends alarms to users and requires user permission in advance. After getting a permit, an attacker can send any content thought notification. When the attacker sends messages through popular services such as Gmail, users can be deceived and click the message.

4.5.2. Autocomplete Input Form Leak. Autocomplete records users' input values per input form ID. When users focus on input form, a Web browser shows autocomplete values that its input ID is same. An attacker sets that the input form width is very narrow and attract users to click it and select autocomplete value. After selecting values, the Web browser sends it to the attacker. An example also looks like a Web-based game to pick users' autocomplete values up [18].

4.5.3. Fake Telephone Call. If call image pops up and a user's phone vibrates, many users might think they got a call. But it could be a fake telephone call. A Web page can pop up an image and vibrate a user's phone, but cannot overlap the status bar of phone, so it looks flimsy. However, vibration deceives users to click the call image or other action [19].

5. Limitation

All browsers do not adopt all elements of HTML5. Attacks mentioned in this paper work or do not work depending on Web browser. The tests came on May 28, 2015. Most attacks work well on the most popular browsers, but some do not. After HTML5 elements are implemented in a browser, the problem will be solved.

Table 2. Dependence on Web Browsers

Web Browsers	IE11	Chrome 43.0	FireFox 38.0	Opera 29.0
Network Scan	X	O	X	O
Keyloggers by SVG	X	X	O	X
Mouse Loggers	O	X	X	X
Web Browsers	Android		iOS(iPhone)	
	Default	Other Web Browsers	Default (Safari)	Other Web Browsers
Vibration Attack	O	O	X	X

In mobile devices, iOS(iPhone) blocks vibration by vibration API. Vibration attacks and fake telephone calls don't work on iPhone.

6. Additional Issues

This section treats additional issues of HTML5.

6.1. Invalid Attacks

These attack scenarios have no effects now.

6.1.1. WebSQL Injection. WebSQL Injection is a variated attack stemming from SQL injection [6]. WebSQL had been proposed as a means for saving data as a DB format in a browser. WebSQL's operation is the same as SQL's. Owing to fairness in the business world, WebSQL was rejected. Certain browsers adopted and implemented WebSQL. WebSQL injection is no longer considered an HTML5 attack.

6.1.2. WebStorage Fill Up. WebStorage Fill Up's target is a user's browser. When a user accesses a Web page, the attacker's code piles up a lot of images or resources onto WebStorage. It exhausts the browser's space and creates a bottleneck for the browser [17]. Nowadays, the browser patches this attack to negate it.

6.1.3. Fake Download. This was part of social engineering. When users chose a file directory, the Web page sent all files from a selected directory. Users intended to select a folder to download a file from a Web page, but the action was opposite. But since the method and browser were changed, this became an invalid attack [20].

6.2. Additional Secure Issues

They are not attack scenarios using HTML5 but can be abused to have harmful effects on users' web browsers.

6.2.1. Phishing with FullScreen API. FullScreen API is not an HTML5 element, and provided by a Web browser's vendor for convenience. Many misunderstood Fullscreen API as an HTML5 element. An attacker can make users mistake an attacker's site for an innocent one. A pop-up message says fullscreen mode; this is social engineering but not an HTML5 attacks [21].

6.2.2. Iframe sandbox attribute and Frame Killer. Frame killer finds an invisible iframe and deactivates it. Iframe sandbox attribute blocks all script in an iframe and is an HTML5 element. If a sandbox attribute exists in a Web page, the frame killer does not work. The two disturb each other's functionality [3].

6.2.3. Evercookie with HTML5. Cookies save short data with an expiration date. Evercookie's role is the same as a cookie's but the former does not expire. This is not an HTML element but developed by a Web developer. Evercookie is not a harmful element, but can be abused for malicious behavior. HTML5 elements can be used to make evercookie. For example, the known methods of making evercookie are canvas or WebStorage [22].

7. Conclusion

This paper described HTML5 attacks, 16 existing and three new. Most attacks affect the use of a Web page or steal information. Many apps and services are developed with HTML5, as well as mobile services, making HTML5 attacks more harmful and destructive.

Web security tends to focus on Web servers, but an HTML5 attack runs on user browsers. After an attack, the end of a tracing attack affects the browser of the victim or user. HTML5 has a variety of dynamic functions, so the number of HTML5 attacks is growing to a critical level.

In our future work, we will study the methods to detect HTML5 attacks and weaknesses.

8. Acknowledgements

This work was supported by the ICT R&D Program of MSIP/IITP. [B0101-15-0230, The Development of Script-based Cyber Attack Protection Technology]

9. References

[1] Ronen Atias, Ofer-Gayer, One of World's Largest Websites Hacked: Turns Visitors into "DDoS Zombies" http://www.incapsula.com/blog/world-largest-site-xss-ddos-zombies.

[2] HTML5 Security Cheatsheet, https://html5sec.org/

[3] TrandLabs, HTML5 OVERVIEW, TrendMICRO, 2011

[4] Xing Jin, Code Injection Attacks on HTML5-based Mobile Apps: Characterization, Detection and Mitigation, Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, 2014

[5] Peter Gramantik, New iFrame Injections Leverage PNG Image Metadata, securiblog, https://blog.sucuri.net/2014/02/new-iframe-injections-leverage-png-image-metadata.html, 2014.02.03

[6] Shreeraj Shah, HTML5 Top 10 Threats Stealth Attacks and Silent Exploits, Blackhat EU, 2012

[7] Paul Stone, Next Generation Clickjacking, Blackhat EU, 2010.

[8] Christian Schneider, Cross-Site WebSocket Hijacking (CSWSH), http://www.christian-schneider.net/CrossSite WebSocketHijacking.html.

[9] Erkkilä, Jussi-Pekka. "Websocket security analysis" Aalto University School of Science (2012): 2-3.

[10] Keylogger Test Page, https://html5sec.org/keylogger/

[11] Vulnerability Summary for CVE-2011-3663, NVD, https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2011-3663

[12] M. Heiderich, Scriptless attacks Stealing more pie without touching the sill, Journal of Computer Security, p.567-599, July 2014.

[13] andlabs, JS-RECONHTML5 based JavaScript Network Reconnaissance Tool, http://www.andlabs.org/tools/jsrecon.html

[14] andlabs, Chrome and Safari users open to stealth HTML5 AppCache attack, http://blog.andlabs.org/2010/06/chrome-and-safari-users-open-to-stealth.html, June 2010.

[15] Kuppan, Attacking with HTML5, Blackhat, 2010.

[16] Weaver, Aaron. "Cross site printing." 2007.

[17] Feross Aboukhadijeh, Introducing the HTML5 Hard Disk Filler™ API,http://feross.org/fill-disk/, Feb 2013.

[18] andlabs, POC for Stealing Auto-Complete Suggestions from Google Chrome, http://www.andlabs.org/hacks/steal_autofill.html

[19] Terence Eden, Malicious Use of the HTML5 Vibrate API, https://shkspr.mobi/blog/2014/01/malicious-use-of-the-html5-vibrate-api/

[20] Krzysztof Kotowicz, Filejacking: How to make a file server from your browser (with HTML5 of course), http://blog.kotowicz.net/2011/04/how-to-make-file-server-from-your.html, 2011.04

[21] Feross Aboukhadijeh, Using the HTML5 Fullscreen API for Phishing Attacks, http://feross.org/html5-fullscreen-api-attack/, 2012.10

[22] CAPEC-464: Evercookie, MITRE, https://capec.mitre.org/data/definitions/464.html, 2012.10.