Code Modification and Obfuscation Detection Test Using Malicious Script Distributing Website Inspection Technology

Seong-Min Park(✉), Han-Chul Bae, Young-Tae Cha, and Hwan-Kuk Kim

Security Industry Technology Division, Korea Internet & Security Agency, Seoul, Republic of Korea
{smpark, hcbae, ytcha, rinyfeel}@kisa.or.kr

Abstract. The use of non-standard plug-ins has been unavoidable in existing HTML. Accordingly, the rising dependence of web content on plug-ins increased and need for additional development suited to platform incurred considerable developmental expenses and time. HTML5 was a suggested solution to this problem, but could cause new security threats through newly added technologies, meaning that web-related elements such as websites, web content, devices and others are exposed to a new type of threat. In this thesis, we suggest a technology to detect website security threats in advance that includes HTML5.

Keywords: HTML5 · Script-based cyber attack · Web content security · Web scanner · Web content analysis

1 Introduction

According to Microsoft, attempted attacks using HTML and JavaScript vulnerability are rapidly increasing. In particular, methods such as web page built-in JavaScript functions, code obfuscation, external JavaScript calling and others have made recent web attacks possible without need for malicious code distribution and infection. Thus detection can be a difficult task with packet-based detection through security devices such as existing web firewalls, IPS and others. Besides, vulnerability to attacks goes up through newly added API and tag after HTML5 standard finalization, and tracking can be difficult since no traces like malicious code are left. And technology to inspect HTML5 security vulnerability is not yet available worldwide [1].

In this situation, we studied a technology to inspect websites that distribute malicious script using HTML5 and analyzed a result tested using such fact. In this thesis, only the core part of the details is summarized, and its composition is as follows. Section 2 examines preceding research on website inspection technology, and Sect. 3 introduces the website inspection technology we studied. Sections 4 and 5 contain our conclusions based on the verification results of the technology we studied.

J.J. (Jong Hyuk) Park et al. (eds.), *Advances in Computer Science and Ubiquitous Computing*, Lecture Notes in Electrical Engineering 421, DOI 10.1007/978-981-10-3023-9_12

2 Related Work

2.1 Security Threat of HTML5

In HTML5, web attacks using new API, tag, and others became possible. Black Hat USA 2012 released HTML5 security vulnerability TOP 10, and researches on HTML5 vulnerability have been released continuously since then [2].

2.2 Web Contents Analyzing Technology

In terms of web contents analysis technology, SpyProxy and WebShield collect traffics in the network level and execute malicious scripts directly, through virtual environment or sandbox based modified virtual browser, and suggest a technology that transmits only malicious scripts eliminated safe contents to client [3].

However, as SpyProxy that collects website information using open source Squid cannot be implemented to virtual environment browser as actual user environment, it makes perfect verification difficult. In addition, for user to use web service, there is a problem of waiting time period until safe contents verification is complete. To resolve such problem, we improved performance using contents split verification technology, but it is quite inconvenient to install additional Agent on host [4].

To resolve problem of behavior monitoring based SpyProxy, WebShield suggests middle box framework creating DOM data structure instead of client browser. Client side Agent is not required. Instead, as webpage data is retained in memory, real time processing becomes possible. However, as result of test, when proxy sandbox occupies 100Mbytes memory, only 82 people per second at the maximum were allowed to have access. In other words, it means there is actual burden that minimum 10 machines are required in enterprise environment. Furthermore, it contains problem of performance deterioration due to User Script handling and frequent DOM (Document Object Model) structure updates, and with the limit to detecting HTML5 based malicious scripts [5].

3 Inspection Technology for Malicious Scripts Distributed Website

As discussed in Sect. 2, because of the performance problem of real time web contents analyzing technology, and limits of prior website inspection tools, new type of frame is necessary to cope with web based attacks. Accordingly, we intend to implement a technology that inspects vulnerabilities by visiting websites before clients use actual services, in the form of a framework.

This structure is divided into website data crawling collection technology, and analyzing technology that inspects collected data. In addition, analyzing technology consists of scripts manipulation inspection, which inspects code modification and obfuscation inspection that extracts original scripts from code obfuscation scripts.

3.1 Website Contents Collection Technology

First step to analyze and prevent website vulnerability in advance, is to collect website contents, an analysis object. Collection technology consists of crawler that performs web contents crawling, and Agent that delivers collection targets, monitors and manages crawler status, and crawler manager that assigns tasks to crawlers, saves collection result into DB.

3.2 Website Vulnerability Analyzing Technology

Inspection is performed for collected website data to figure out if there are vulnerabilities through website vulnerability analysis technology. Largely, 2 types of inspections are carried out which include scripts obfuscation inspection and modification inspection.

Attacker launches attack using scripts obfuscation for malicious website scripts not to be detected [6]. Therefore, obfuscation script should be identified and de-obfuscation needs to be carried out. First, to inspect obfuscation status of script code, measure Entropy element, N-Gram Entropy element, Max Word Size element, and then calculate obfuscation score [7].

Entropy element is to measure distribution status of bytes within web documents, and expression is as follows (b = count of each byte, T = total count of bytes, N = count of strings)

$$E(B) = \sum_{i=1}^{N} \left(\frac{b_i}{T}\right) \log\left(\frac{b_i}{T}\right) \begin{cases} B = b, i = 0, 1, \ldots, N \\ T = \sum_{i=1}^{N} b_i \end{cases} \tag{1}$$

N-Gram Entropy element measures distribution status of special characters within web documents, and then the ratio to entire Entropy.

$$R(S) = \frac{E(S)}{E(B)} * 100 \tag{2}$$

Max Word Size element measures length of the longest character string within web documents, and if each measured value exceeds given tolerance, final obfuscation score is calculated by adding up score of each element.

Obfuscation code was decoded by executing obfuscation scripts classified in such way, with V8 JavaScript engine. And we determined malicious behavior by extracting web page scripts calling data through API hooking.

Second, script modification inspection verifies if scripts are manipulated to cause malicious behaviors. If collected web contents are HTML based documents, DOM data is generated and scripts extracted. Hash value from extracted DOM data or JS file is extracted and saved. Non-cryptographic hash algorithm such as MurmurHash, CRC32, SuperFastHash, and others is used for hash value extraction. In general, while web page contents change frequently, scripts do not change unless corresponding website is

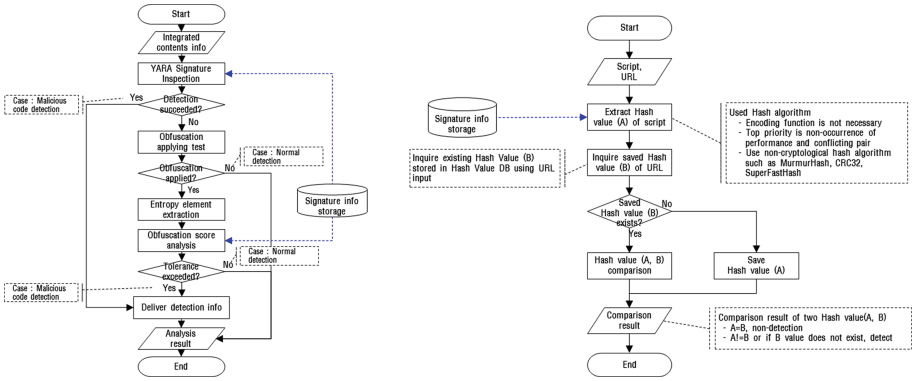

Fig. 1. Script obfuscation inspection algorithm and Scripts modification inspecting algorithm

restructured. Therefore, once hash values are saved in URL unit, based on the value as a reference, it can be compared with the hash value extracted when next user visits the same URL. As result of comparison, if hash value is the same, modification did not occur, and if different, it is regarded as modification, and static analysis is conducted (Fig. 1).

4 Code Modification and Obfuscation Detection Test

We tested the inspection technology for malicious scripts distributing website, which is implemented using 17 types of attack samples. Samples are made up to implement attack behaviors such as Scanning, Hijacking, DDoS triggering, data intercepting, and others, by injecting malicious scripts in websites. After injecting each malicious script into websites implemented for the test, we executed one test at a time. When accessing test websites from tester, it should go through inspection system that developed technology is applied, in order to verify detection process. In addition, in terms of each malicious script type, we tested total 4 types of various attack mechanisms by applying 2 types of code modification methods and 2 types of obfuscation methods (Table 1).

As result of test, 100 % detection rate was shown for original code of each attack script, but for codes that applied variable name or function name during code modification, 58.8 % of low detection rate was shown. However, with 100 % detection rate for code modified by applying Code Split method and obfuscation code applied with Packed encode method, we could confirm that it is detected being dependent upon signature ID.

Secondly, we conducted test on the performance of implemented inspection technology with commercial website as subject. Designated as safe websites by Google Safe Browsing and Norton Safe Web among Alexa Top 100 websites, 50 websites were selected. In addition, test was conducted using 50 safe websites such as Amazon, E-bay, Alibaba, and others, and 10 websites converted to HTML5 operating in Korea as subject [8].

Number of URLs that can be collected as test measurement items, and number of misdetections were used as subject. Depth5 level crawling was performed by inspection

Table 1. 17 types of malicious script samples used in test.

Sample No.	Attack	Description
1	Network Scan	Without user knowing, confirm user's internal network or port open status by using XHR(XML Http Request), WebSocket, WebRTC
2	Port Scan	
3	Cross Site WebSocket Hijacking	Without user knowing, execute WebSocket communications and establish connection with WebSocket server
4	ClickJacking (Drag&Drop)	Induce user's click or user's file (using Drag-Drop API, File API) to other places by overlapping transparent frame set as opacity (transparency) property 0, with other frame
5	ClickJacking (Click info)	
6	WebSocket data intercepting	Intercept WebSocket communication details by redefining onmessage event handler of WebSocket
7	SVG Keylogger	Send request to attacks to attacker based on user's key input through accessKey event of SVG(Scalable Vector Graphics), a XML type image
8	MouseLogger	Obtain touch event coordinates of user mouse using Pointer API
9	Cross Site Printing	Induce to print GET message by sending it to basic port 9100 of printer using XHR, WebSocket
10	Cookie Sniffing	Without user knowing, transmit the cookies within browser to attacker
11	Geolocation	Figure out device location using Geolocation API and send to attacker
12	Server-Sent Event Bot	Insert to scripts and modify web pages using Server-Sent Event, the API that provides Polling function
13	Worker DDoS	Without user knowing, trigger DDoS using Web Worker that provides separate JavaScript threads function in web page
14	WebStorage Leak	Inquire Web Storage information and transmit externally
15	IndexedDB Leak	Intercept WebSQL details which is managed by each domain within browser (However, WebSQL was eliminated from standard, and instead, Indexed DB is selected as standard)
16	Vibration Attack	Trigger user inconveniences by causing continuous vibrations in web pages
17	Script DoS (for statement)	Send request externally whenever web page connection occurs by putting scripts that send request externally into web pages

subject website, and we were able to measure number of URLs collected per second. Number of entire URL collection was 33,698, with 15,579 s required. Number of URLs collected per second based on such fact was 2.16 EA, and URL collection capacity of 1 crawler per day was measured as total 186,887 EA (Table 2).

Table 2. Performance index.

Item	Result
Number of total URL collected	33,698
Total time spent (sec)	15,579
Number of URL collected per (URL/s)	2.16
Number of URLs collected per day (1 crawler)	186,887

In addition, we conducted analysis on misdetection results along with detection results on subject sites. Number of total URLs, attempted in the analysis was 10,464 EA, 87 cases of them were confirmed as misdetection. Most of detection items confirmed as misdetection were Clickjacking attack by signature Rule ID No. 11. Clickjacking is a type of attack that sets opacity property in scripts to 0, and induces user click to malicious scripts by overlapping transparent frame with other frame. As result of applying sample HTML document about Clickjacking to YARA tool, we were able to confirm that most of detections were due to script jquery.js. Opacity property is set when suing jquery.js, and thus misdetection was occurred.

5 Conclusion and Future Work

This thesis suggests a technology to collect and analyze contents such as HTML, SCRIPT, CSS, and others, in order to conduct inspection of websites distributing malicious scripts including HTML5. Inspection test including attack samples detection, and commercial websites was conducted. In terms of performance, this technology is differentiated from existing real time malicious scripts detection technology such as SpyProxy and WebShield. In enterprise environment, when intending to collect and analyze one million web contents per day, WebShield requires 10 machines at the minimum, but with this technology, far less number of machines are needed to achieve the same goal. As number of URLs collected by 1 crawler is 186,887 EA, only 6 (5.35) machines is required to collect 1 million URLs per day.

However, some improvement needs to be made for part of test process. In commercial network website test, dynamically generated web pages were not collected during collection process using web crawling tools, instead of direct visit. Furthermore, collection was not possible due to IP blocking by web service providing company. To resolve such problem, by granting 1 s interval for each collection depth, number of URLs collected per second was reduced more than expected.

Acknowledgments. This study was conducted as part of information communications and broadcasting R&D project by Ministry of Science, ICT and Future Planning, and Institute for Information & Communications Technology Promotion [B0101-15-0230, Prevention of script based cyber attack and development of counter technology].

References

1. Microsoft Security Intelligence Report (SIR) Volume 17, pp. 43–54. Microsoft (2014)
2. Shah, S.: Founder & Director, Blueinfy Solutions, “HTML5 Top 10 Threats Stealth Attacks and Silent Exploits”, pp. 1–20. Black Hat, USA (2012)
3. Agten, P., Van Acker, S., Brondsema, Y., Phung, P.H., Desmet, L., Piessens, F.: Jsand: complete client-side sandboxing of third-party javascript without browser modifications. In: Proceedings of the 28th Annual Computer Security Applications Conference, pp. 11–21. ACM (2012)
4. Moshchuk, A., et al.: SpyProxy: Execution- based Detection of Malicious Web Content. USENIX Security (2007)
5. Li, Z., et al.: WebShield: enabling various web defense techniques without client side modifications. In: NDSS (2011)
6. Xu, W., Zhang, F., Zhu, S.: The power of obfuscation techniques in malicious JavaScript code: A measurement study. In: 7th International Conference on Malicious and Unwanted Software (MALWARE). IEEE (2012)
7. Choi, Y., Kim, T., Choi, S., Lee, C.: Automatic detection for javascript obfuscation attacks in web pages through string pattern analysis. In: Lee, Y.-h., Kim, T.-h., Fang, W.-c., Ślęzak, D. (eds.) FGIT 2009. LNCS, vol. 5899, pp. 160–172. Springer, Heidelberg (2009). doi:10.1007/978-3-642-10509-8_19
8. Alexa Top 100. http://www.alexa.com/topsites