Real-Time Malicious Script Blocking Technology at the Host-Level

SangHwan Oh, HanChul Bae, Seongmin Park, HwanKuk Kim (Author)
Security R&D Team 2
Korea Internet & Security Agency
Seoul, Korea
osh1, hcbae, smpark, rinyfeel@kisa.or.kr

Abstract. Due to the diversity of mobile devices, interests have been increased towards HTML5, the next generation's web standard which pursues cross platform. To play media files or process 3D graphics in previous HTML environments, users had to install non-standard plug-ins such as Silverlight or Active X. On the other hand, HTML5 provides new tag functions of audio, video etc and new java script functions of Websocket, Geolocation API etc to substitute non-standard technologies such as Active X. To use such functions of HTML5, web browser developers are competitively applying HTML5 to their browsers, making the active conversion to HTML5 a global trend of today. Along with such trend, however, the risk of new cyber attacks taking advantage of java scripts, the key function of HTML5, is also increasing. Cyber attacks based on scripts can trigger vicious actions when the user just accesses web pages inserted with vicious scripts, and thus there are limits in detection using previous security technologies. This paper proposes a technology which collects and analyzes HTTP traffic generated through web browsers at host level to detect and block vicious scripts.

1 Introduction

Services provided through the online environment are growing more diversified due to the fast dissemination of the Internet. Traditional HTML could provide only static services, and non-standard plug-ins like Active X were required to provide dynamic service, which can cause security. The most powerful method to resolve the issue is the next-generation web standard HTML5, which was announced in October 2015. [1] Various functions were added to HTML5 that could replace non-standard plug-ins but retained compatibility with traditional HTML. For instance, HTML5 controls media using new tags: video, audio and Canvas that replace Adobe Flash. Moreover, it enables implementation of multi-thread (Web Worker), web socket communication and utilization of location information by using additional JavaScript APIs. In addition, HTML5 supports cross-platform, and Gartner has selected HTML5 as one of its top 10 mobile technologies. [2] In line with the trend, the world's leading web browser developers are rushing to upgrade their browsers to support HTML5.

L. Barolli et al. (eds.), *Advances on Broad-Band Wireless Computing, Communication and Applications*, Lecture Notes on Data Engineering and Communications Technologies 2, DOI 10.1007/978-3-319-49106-6_47

Google has set its policy to make mandatory use of an HTML5 player to play video on its browsers, Chrome and FireFox. In such an environment in which conversion to HTML 5 becomes mandatory, however, security threats to HTML5 are increasing. XSS (Cross Site Scripting) attack using new tag uses video and audio that can pass through XSS attack detection filter, which uses pattern matching. In particular, JavaScript, the core function of the new features, is facing more threats. The cyberattack on the Korean government and public institutions on June 25, 2013, was an example of a "script-based cyberattack" using JavaScript. Additionally, the script-based attacks took place in the Chinese video sharing site SOHU TV in 2014 and the open source share site GitHub in 2015. In a traditional malicious code attack, the codes were required to be downloaded and installed on a user PC to perform malicious behavior and existing detection technologies could detect and block those malicious codes. For a script-based cyberattack, however, since JavaScript is running on web browsers only, once the browser is closed, no trace remains on a user PC, which makes difficult for existing security technologies to deal with it. And since most JavaScript is obfuscated to protect developers' ideas or improve performance, attackers also obfuscate their malicious script, which can easily pass through traditional signature-based static detection technology. Thus this study describes technology to detect a script-based cyberattack at the host level. Chapter 2 describes earlier research on technologies to detect web attacks, Chapter 3 features technology to detect and block malicious script through local proxy, Chapter 4 analyzing the result of applying the processed technology, and Chapter 5 presents the conclusion and future direction of research.

2 Paper Preparation

Script-based cyberattack is web-based attach which is made through web. Responsive method to the web-based cyberattack can be mainly categorized into two; network level and client level method. The paper describes client level method which is related to this study, and JavaScript obfuscation technology which is one of the biggest issue in script-based cyberattack.

Client level technology

There are currently a number of on-going researches on client level technologies to respond to web-based cyberattack. The most responsive technology among those is one used in WebCheck[3], SiteAdvisor[4], etc that inspects URLs connected on browser. The technology blocks access to malicious web sites by matching URL to access and black list of URLs with history of malicious code distribution. However, there is still difficulty to deal with the attack through only this method as attackers frequently change their URLs.

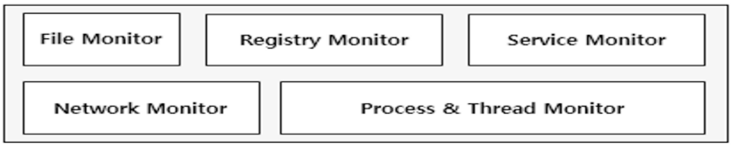


Fig. 1. Configuration of Observer

To come over with the limits, technology to monitor web browser has been developed as it is start-point of attack. [5] As show in [Fig 1] the technology suggested method using malicious behavior detection module called Observer. As show in [Fig 2] It applies process unit in browser via Observer module consisting of File Monitor, Network Monitor, Process & Thread Monitor, etc and collects events from the processes via API hooking to detect malicious behavior. As monitoring web browser itself, while the technology is useful to respond to web-based attack, it makes high loads as it creates Observer and monitors all processes in browser. On the other hands, the technology requires high specification client device which is major weak point in client level which should be operated in various environments.
Therefore, the paper describes technology with light in accordance with client environment, but powerful detection function.

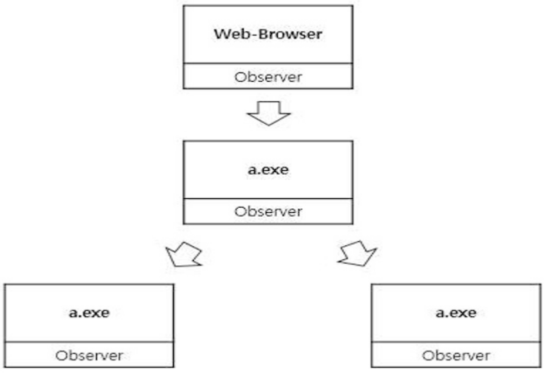


Fig. 2. Observers attached to the each of the processes

JavaScript obfuscation

Along with diversification of web-based cyberattack, various technologies responding to the attack have been developed. The most frequently used technology is signature-based static analysis technology which extracts pattern of malicious codes and analyzes them via pattern matching with analysis target. Attackers are trying to pass through the pattern matching based static analysis technology by generating new malicious codes via obfuscation.
In script based attack, they evade signature-based method by obfuscating JavaScripts in various ways. The most common methods are string split which splits malicious code into a number of string variables and combines them when executing and method to convert character set to ASCII code by using JavaScripts' own functions

(escape, and unescape). IN addition to those methods, attackers try to disarm the existing security technologies by using XOR encoding or applying their own encoding functions. Therefore, the study provides method to de-obfuscate the JavaScripts.

3 Proposed Technology

As show in [Fig 3], the technology proposed in this paper is composed by a collection module realized in the form of Local Proxy at host level, and an analysis agent module which decides viciousness and sets the processing policy afterwards. This technology is installed and operated in user PC to collect and analyze HTTP packets generated in web browsers. When decided to be vicious, the technology protects user PC from script-based attacks by deleting the vicious script or by using post-processing methods such as redirecting to a safe page etc.

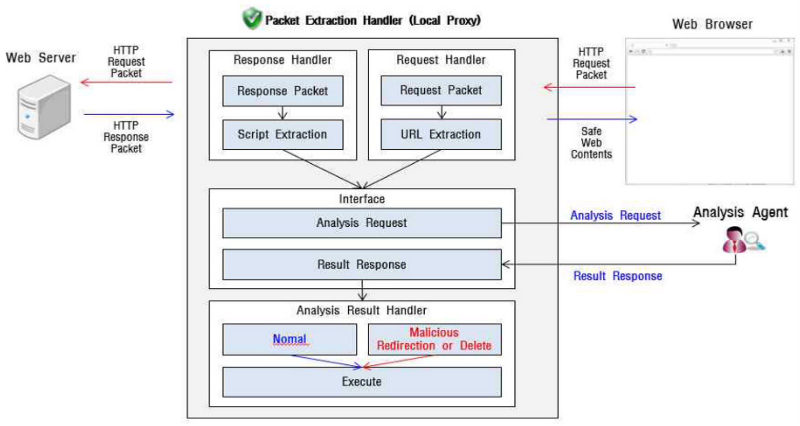


Fig. 3. Propsed Technology architecture

Packet collection module

First, as the collection module for the subjects of analysis, Tinyproxy, a famous light-weight proxy daemon among open-source SWs, was used. [6] To collect HTTP packets using Local-Proxy, registry value related to the proxy setup of Windows was changed as [Table 1]. By doing so, all packets generated from web browsers were received and transmitted through the collection module.

Table 1. Proxy settings related Windows Registry values

Windows Registry Values
HKEY_CURRENT_USER₩Software₩Microsoft₩Windows₩CurrentVersion₩ProxyEnable
HKEY_CURRENT_USER₩Software₩Microsoft₩Windows₩CurrentVersion₩ProxyServer

By collecting HTTP Request packets sent to external web server from the web browser of user, access URL information is extracted. Then, request is made to the analysis agent. If the analysis agent decides that the relevant URL is safe, the packets are sent to the web server. Afterwards, script codes are extracted in tag units from HTTP Response packets sent by the web server. Among the extracted scripts, external scripts where script codes are located outside using src property etc are sent to web browser, so that the browser can request again the relevant script. By doing so, external script codes are extracted from the Response packets. Script codes collected as above are requested to the analysis agent for analysis in tag units. Based on the result of analysis, packets which went through post-processing are sent from the analysis agent to the web browser to protect the web browser of user PC from vicious scripts.

Analysis agent module

As shown in [Fig 4], the analysis agent module mainly performs real-time analysis and precise analysis. Real-time analysis is divided into URL test and static analysis of scripts. Viciousness is decided based on the blacklist of access URL [5] extracted from HTTP Request packets generated in web browser. If decided as vicious URL, information on the access to vicious URL is noticed to user. If the user blocks the access, the module redirects to a safe page and finishes the analysis. If the URL is normal, HTTP Request packets are sent to the relevant web server, and scripts are extracted from HTTP Response packets received from the web server to conduct static analysis of scripts.

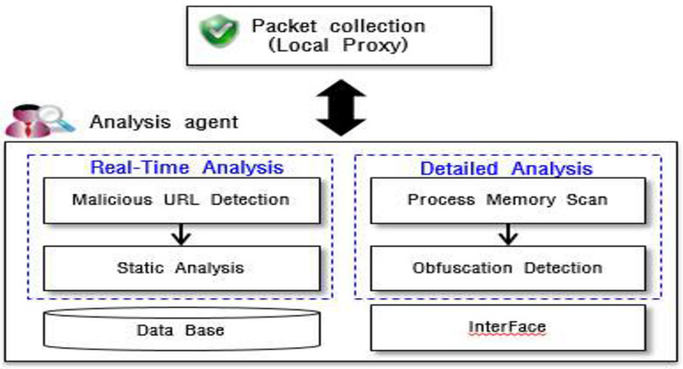


Fig. 4. Analysis Agent Module

Static analysis decides viciousness through signature pattern matching based on YARA[7]. YARA is a technology which searches and classifies vicious codes based on character strings or binary patterns. Its most significant characteristic is the ability to generate Rule Sets which can contain all kinds of expressions. Using YARA Rule, the pattern information of vicious scripts' codes generated before is extracted and used as signature for static analysis. Script codes extracted from HTTP Request packets go through pattern matching based on YARA Rule to decide for the existence of vicious scripts. However, since obfuscated vicious scripts change the patterns themselves into other character strings, it is difficult to detect them using static analysis based on signature pattern matching. To detect such obfuscated vicious scripts which can detour static analysis, the technology of precise analysis has been additionally introduced. Precise analysis is divided into the first step of memory scan which extracts scripts where obfuscation is removed and the second step of obfuscation analysis where obfuscation is removed to extract the original. Memory scan extracts PID of web browser which is currently running, and scans the memory of the relevant browser process to extract scripts currently running in the browser. At this point, scripts where obfuscation is removed can be secured as the first step. Scripts where obfuscation is not removed from the first step are applied with V8 java script interpreter [8] used in Chrome Browser to remove such obfuscation. After running the obfuscated scripts on V8 interpreter, BreakPoint is set at a certain point to extract script codes. By doing so, script codes where obfuscation is removed can be secured. Through signature pattern matching used in static analysis, these script codes are analyzed to see if they are vicious.

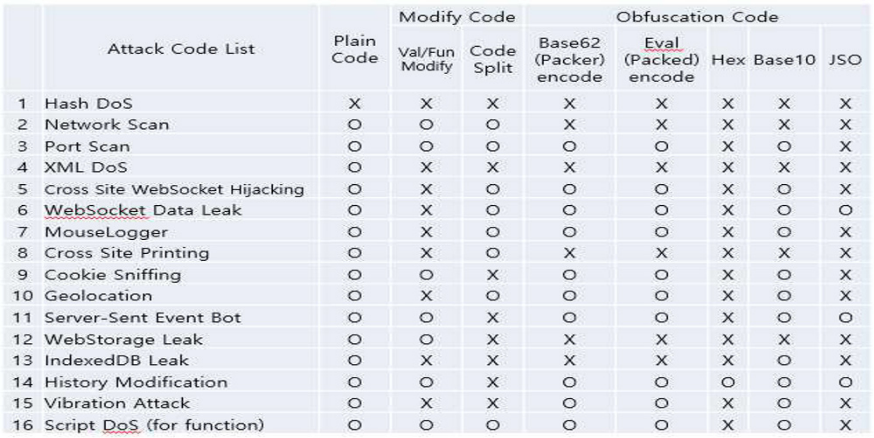
4 Analyzing the result of applying the proposed technology

In order to check the real-time static analysis of script based on YARA Rule, a technology proposed in this paper, as well as the decryption function for obfuscated script using V8 java script interpreter, the detection in 128 samples was checked. These 128 samples were made by applying eight changes such as code division, change of function name, obfuscation etc to 16 types of vicious script attack samples.

Real-time static analysis of script

Static analysis of script determines viciousness by matching major keywords in vicious script code based on YARA Rule. Therefore, detection was checked in 16 original attack samples and 112 samples where detours can be made by changing keywords based on the change of function/variable names. As shown in [Table 2], the detection rate in original samples was more than 95% since the major keywords in vicious scripts were all included in the scripts subjected for analysis. However, in samples which detoured pattern matching through the change of function/variable names, code division, obfuscation etc, the detection rate was less than 50%. In case of changing function/variable names, keywords themselves were changed to detour pattern matching, and in case of code division, detection was difficult since vicious keywords were divided into two or more script tags while the technology of this paper performs analysis in the unit of script tag (<script>…</script>). In case of obfuscation, high detection rate was shown for Base62, Base10, Eval function encoding since there was no change on the keywords themselves. However, for obfuscation samples such as Hex or JSO where the keywords were changed, detection failed.

Table 2. Result of Static Analysis



	Attack Code List	Plain Code	Modify Code		Obfuscation Code				
			Val/Fun Modify	Code Split	Base62 (Packer) encode	Eval (Packed) encode	Hex	Base10	JSO
1	Hash DoS	X	X	X	X	X	X	X	X
2	Network Scan	O	O	O	X	X	X	X	X
3	Port Scan	O	O	O	O	O	X	O	X
4	XML DoS	O	X	X	X	X	X	X	X
5	Cross Site WebSocket Hijacking	O	X	O	O	O	X	O	X
6	WebSocket Data Leak	O	X	O	O	O	X	O	O
7	MouseLogger	O	X	O	O	O	X	O	X
8	Cross Site Printing	O	X	O	X	X	X	X	X
9	Cookie Sniffing	O	O	X	O	O	X	O	X
10	Geolocation	O	X	O	O	O	X	O	X
11	Server-Sent Event Bot	O	O	X	O	O	X	O	O
12	WebStorage Leak	O	O	X	X	X	X	X	X
13	IndexedDB Leak	O	X	X	X	X	X	O	X
14	History Modification	O	O	X	O	O	O	O	O
15	Vibration Attack	O	X	X	O	O	X	O	X
16	Script DoS (for function)	O	O	O	O	O	X	O	X

Decryption of obfuscated script

By decrypting obfuscated scripts using V8 java script interpreter, the extraction of original scripts was checked. Most of the original scripts were extracted for Base62, Base10, Eval function encoding samples which used exclusive tools for obfuscation. However, obfuscation methods such as Hex or JSO were impossible to decrypt. Different from previous methods, Hex or JSO has been confirmed to apply two or more obfuscation methods, making V8 java script interpreter proposed by this paper impossible to extract original codes at BreakPoint.

5 CONCLUSION

Due to the diversity of mobile devices such as laptop PCs, tablet PCs etc, interests have been increased towards HTML5 which pursues cross platform. Along with such increased interests, however, the risk of cyber attacks taking advantage of reinforced java scripts is also increasing. This paper proposed a technology that can deal with cyber attacks taking advantage of java scripts in user PC. The technology can detect cyber attacks based on scripts which are difficult to detect using previous security solutions. The paper also proposed detection methods for obfuscated scripts that detour security technologies based on pattern matching. The technology proposed in this paper can be used to increase detection rate by linking with detection technologies for vicious codes proposed previously. Also, by combining the technology with AP equipments, AP with security functions can be created. Still, the environment of technology proposed in this paper is limited to user PC, and there are also improvements to be made on the analysis technology. First, to prepare for the era of HTML5 which pursues cross platform, technological researches on detecting vicious scripts in mobile environment should be conducted. Also, as the analysis of the result of applying the proposed technology suggests in Paragraph 4, researches should be conducted to use URL instead of script tag as analysis unit so that vicious scripts divided onto two or more tags can be detected. Finally, regarding the analysis of obfuscation, codes using well-known obfuscation tools were decrypted, while codes applied with two or more obfuscation methods were difficult to decrypt. In the future, we are planning to solve the issue of obfuscation by studying algorithms related to decryption.

Acknowledgments. The study has been conducted as a part of ICT R&D project sponsored by Ministry of Science, ICT and Future Planning and Institute for Information & communications Technology Promotion. [B0101-15-0230, Development of preventative and responsive technologies for script based cyberattack]

References

1. W3C, “HTML5 Standard”, April 20 2015, http://www.w3.org/standards
2. Gartner, ″Gartner Identifies Top 10 Mobile Technologies and Capabilities for 2015 and 2016″, February 24 2014.
3. KISA, WebCheck. Available : http://webcheck.kisa.or.kr
4. McAfee. SiteAdviser. Available : http://www.siteadvisor.com
5. Young-wook Lee, Dong-jae Jeong, Sang-hoon Jeon, and Chae-ho Im, ″Design and Implementation of Web-browser based Malicious behavior Detection System(WMDS)″ Journal of Information Security & Cryptology June, 2012
6. Tinyproxy, https://tinyproxy.github.io/
7. YARA Documentation, http://yara.readthedocs.org/en/latest/index.html
8. Chrome V8, https://developers.google.com/v8