J Inf Process Syst, Vol.12, No.3, pp.422~435, September 2016
http://dx.doi.org/10.3745/JIPS.03.0045

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

ELPA: Emulation-Based Linked Page Map Analysis for the Detection of Drive-by Download Attacks

Sang-Yong Choi*, Daehyeok Kim**, and Yong-Min Kim***

Abstract

Despite the convenience brought by the advances in web and Internet technology, users are increasingly being exposed to the danger of various types of cyber attacks. In particular, recent studies have shown that today's cyber attacks usually occur on the web via malware distribution and the stealing of personal information. A drive-by download is a kind of web-based attack for malware distribution. Researchers have proposed various methods for detecting a drive-by download attack effectively. However, existing methods have limitations against recent evasion techniques, including JavaScript obfuscation, hiding, and dynamic code evaluation. In this paper, we propose an emulation-based malicious webpage detection method. Based on our study on the limitations of the existing methods and the state-of-the-art evasion techniques, we will introduce four features that can detect malware distribution networks and we applied them to the proposed method. Our performance evaluation using a URL scan engine provided by VirusTotal shows that the proposed method detects malicious webpages more precisely than existing solutions.

Keywords

Drive-by Download, Malware Distribution Network, Webpage Link Analysis, Web Security

1. Introduction

Today's cyber attacks have evolved from direct attacks on physical systems to a more sophisticated form of Internet-based ones, which implies that any web service or user can be a target of attackers. Specifically, attackers compromise a legitimate website to spread a malicious code to website visitors, and this type of attack is so called a drive-by download attack [1-4]. Recent studies [5-8] have shown that a drive-by download is the most prevalent type of attack amongst all Internet-based threats.

To accomplish a drive-by download attack, attackers try to redirect visitors of compromised websites to a malware distribution page. Thereby, a victim automatically visits the chain of webpages, which is known as a Malware Distribution Network (MDN). This attack would bypass existing detection methods by using JavaScript obfuscation and a hiding technique as well as exploiting the zero-day vulnerabilities of software. This attack can lead web users to serious threats, such as personal information leakage, while an infected PC can be utilized for a large-scale cyber attack, such as

Manuscript received January 28, 2015; accepted August 25, 2015; onlinefirst December 31, 2015.
Corresponding Author: Yong-Min Kim (ymkim@chonnam.ac.kr)
* Cyber Security Research Center, Korea Advanced Institute of Science and Technology, Daejeon, Korea (csyong95@gmail.com)
** School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, Korea (dhkim7@kaist.ac.kr)
*** Dept. of Electronic Commerce, Chonnam National University, Yeosu, Korea (ymkim@chonnam.ac.kr)

Distributed Denial-of-Service (DDoS).

Identifying a distribution site and blocking them in advance is one of the best ways to prevent a drive-by download attack. The other way to prevent this is removing the vulnerability of the website and protecting it from being compromised. First, to effectively detect a MDN, various methods have been proposed, and they can be classified into two categories. One is based on static analysis [9-11], such as pattern matching and metadata analysis; the other is the dynamic analysis method [12-14], which includes script emulation and virtual machine-based validation. However, existing works have limitations when it comes to recent sophisticated evasion methods, such as obfuscating, hiding, and circumventing the virtual machine environment. Second, to detect the vulnerability of the website, various tools have been proposed [15,16]. These tools are for analyzing the vulnerability of an application (or plug-in), such as PDF and FLASH, which are included in the website. However, the focus of our study is to more effectively identify a MDN that has already been created by attacker. Therefore, in this study we used the approach to analyze the features of the malicious website.

In this paper, we present an effective method for detecting recent advanced drive-by download attacks. The proposed method does not simply match the pattern of the contents, but analyzes the behavior of malware distribution. To analyze the connection of webpages used for malware distribution, it extracts the structure of the MDN using script emulation. We present four features of the MDN to detect a malicious connection of webpages. We generated these features from the analysis result of a collected dataset of a MDN and utilized features used in existing works.

We evaluated the detection performance of the proposed method by comparing it with VirusTotal [17], a URL scanning engine. The evaluation result shows that our method outperforms the existing detection method in terms of detection rate.

2. Related Work

2.1 Drive-by Download Attack

In a drive-by download attack, attackers compromise legitimate websites with web attack methods, such as SQL injection, and make them landing pages of the MDN. They manipulate the websites to covertly redirect visitors to MDNs by adding related JavaScript or HTML tags. To maintain the stealth of the attack, attackers use a hidden frame (or iframe) and compromised advertisement banner or 3rd party widget [1]. Note that a set of connected webpages used in a drive-by download attack is a MDN, which generally consists of a landing page, a set of hopping pages, and a distribution page. In the attack, once the victim visits a compromised website (landing page), he or she is automatically redirected to a malware distribution page via some hopping pages.

2.2 Detection Methods and Limitations

Various detection methods against drive-by download attacks have been proposed and they are classified into two categories: static analysis and dynamic analysis. First, static analysis utilizes information on the source codes of webpages or the metadata of websites.

Table 1. Characteristics of methods for detection of drive-by download attacks

Category	Method	Characteristic
Static analysis	Pattern matching	Fast, but cannot precisely analyze obfuscated contents
	Metadata analysis	Statistical analysis with relatively low precision
Dynamic analysis	Browser emulation	Fast compared to VM-based method Higher precision compared to static analysis Malware can detect and evade emulation environment
	Virtual machine based method	Analyzing the changes in system during the download Malware can response differently depending on the execution environment.

[start] : Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0;)
var $c=new Date();
$c.setTime($c.getTime()+30*24*3600*1000);
document.cookie="tigerinfo_ieversions8=update;expires="+$c.toGMTString();
</script>
[end] : Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0;)

[start] : Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0;)
var $c=new Date();
$c.setTime($c.getTime()+30*24*3600*1000);
document.cookie="tigerinfo_ieversions8=update;expires="+$c.toGMTString();
</script>
<script>document.write("<iframe src=http://www.fgucny.com/inc/top.html width=100 height=1 frameborder=0></iframe>");</script>
[end] : Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0;)

Fig. 1. Different HTTP responses depending on web-browser version.

Specifically, the pattern matching method [9,13,14] based on JavaScript functions or character strings, which are widely used for distributing malicious codes, have been proposed. In addition, the metadata of websites or the server, such as the URL of webpages, an IP address, and location of the hosting server, is statistically analyzed for detecting malware distribution [10-12].

Second, dynamic analysis methods basically utilize the virtual machine environment or web browser emulation. In virtual machine based methods [12,13], they visit websites with the vulnerable system setting (e.g., using an unpatched version of Internet Explorer and Windows OS) and check whether any malicious changes have occurred in the system. On the other hand, the emulation-based dynamic analysis method emulates the web browser engine to analyze the behavior of website that a user has visited [14].

Table 1 shows the summary of existing detection methods and their characteristics. It is important to note that existing methods have limitations against evasion techniques, such as JavaScript obfuscation, and making different HTTP responses depending on the requesting environment. JavaScript

obfuscation can be achieved by using the escape function, eval() function, customized encoding, 8-bit ASCII encoding, Base64 encoding, or XOR encoding [18].

Moreover, malicious or compromised websites send visitors different contents based on the HTTP request message. Fig. 1 shows a real world example where a website only responds with a malicious link if the visitor is using Internet Explorer 8.

3. ELPA: Emulation-Based Linked Page Map Analysis

3.1 Challenges for Detecting a Malware Distribution Network

There are three key challenges to precisely detecting and analyzing drive-by download attacks. First, an analysis method should be able to analyze JavaScript that uses an obfuscation technique to find various forms of hidden links connected to a hopping page, thereby extracting the entire structure of MDNs. Second, the method should support and analyze using a multiple connection environment to deal with the evasion method, which makes spurious HTTP responses. Lastly, the performance of the detection method should not be affected by an attacker's evasion methods.

We resolved the first and second challenges by using web browser emulation [19], which originates from an open-source JavaScript emulator called SpiderMonkey [20]. The emulator is able to fully trace a chain of links, which constructs a MDN by completely emulating various types of syntax in HTML and JavaScript related to automatic redirection. Also, it supports numerous versions of browsing environments to incapacitate the evasion technique (Refer to [19] for a detailed description of the emulator we used).

In this paper, we focus on the third challenge. We introduce the concept of a Linked Page Map (LPM), which illustrates the automatic redirection in the MDN. We propose an Emulation-based Linked Page Map Analysis (ELPA), which analyzes LPM with a set of features and decides on its maliciousness.

3.2 Emulation-Based LPM Analysis (ELPA)

3.2.1 Linked Page Map

We define a Linked Page Map (LPM) as a chain of webpages potentially regarded as a MDN. A LPM is represented with a directed graph, as shown in Fig. 2. A LPM consists of nodes and directed links. Each node indicates a webpage and a directed link exists between two pages if an automatic redirection can occur. There can be multiple paths via different landing and hopping pages to the same exploit page.

Each node is represented as a tuple with 6 attributes as follows:

$$\mathrm{LPM}(i),(j)=\{\text{Nodename, Domainname, Linkatr, Nodeidx, Pnodeidx, Snodeidx}\} \quad (1)$$

where $i,j \in N$

- Nodename: URL currently being analyzed

- Domainname: Domain name in the URL
- Linkatr: Link between
- Nodeidx: Node ID
- Pnodeidx: Node ID of a parent node, NULL if an landing page
- Snodeidx: Node ID of a child node, NULL if an distribution page

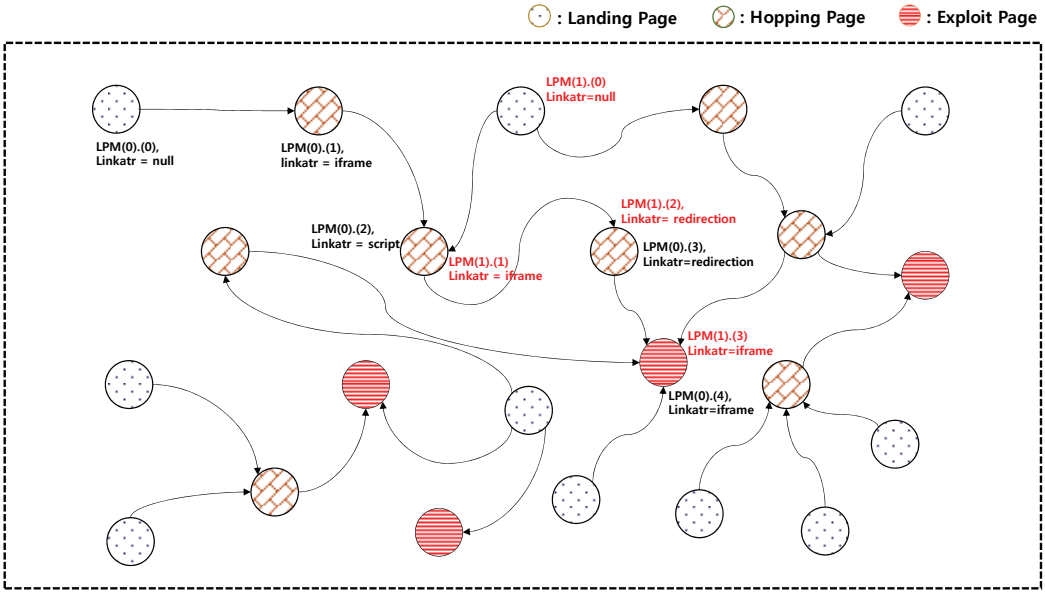

Fig. 2. Concept of Linked Page Map (LPM).

3.2.2 Proposed features

Before we introduce the LPM features that ELPA uses for detecting MDNs, we will summarize the MDN features presented by previous works in Table 2. We classified the existing features into four categories: DOM information, statistical information, behavioral information, metadata, and malicious information. The three columns (contents, information, etc.) on the right of the table indicate which element of the webpage has to be analyzed for extracting each feature. For example, analyzing 'contents' means that a detection method analyzes the source code of the webpage, such as the JavaScript function and the strings used for constructing the webpage. Moreover, to extract metadata, including the AS number of hosting server and connection speed, needs to analyze the "information" of the webpage needs to be analyzed. Also, malicious information would be related to previously analyzed results and other history.

We statistically analyzed 7,390 LPMs (3,437 MDNs and 3,953 non-MDNs) that were previously collected and selected existing features that are less affected by attackers and are closely related to an indication of malware distribution. Also, we defined four new features, which will be described in detail in the next subsection. Table 3 shows the LPM features used by ELPA.

Table 2. Features of related-work for detection of Malware Distribution Network (MDN)

Existing work	Category	Feature	Subject to be analyzed		
			Contents	Information	Etc.
JSAND [14]	DOM	Length of dynamically evaluated codes	O		
		Sequence of method calls	O		
	Statistical	Browser personality and differences		O	
		Number and target of redirection	O		
		Number of dynamic code execution	O		
		Ratio of string definitions and uses	O		
		Number of likely shellcode strings	O		
		Number of instantiated components	O		
	Behavior	Values of attributes and parameters in method calls	O		
		Number of bytes allocated through string operations		O	
Safebrowsing [12]	Metadata	Hosting Info		O	
		Site status	O		
		Script S/W version		O	
	Maliciousness	Maliciousness of webpage			O
		Compromised by an advertisement			O
SpyProxy [13]	DOM information	HTML element related to malicious operation	O		
URL learning [10]	Metadata	AS number		O	
		Connection speed		O	
		Geographic		O	
		IP prefix		O	
		Whois Info		O	
		Hostname	O		
		Last path tokens	O		
		Path tokens	O		
		Primary domain	O		
		Top level domain	O		

Table 3. Features of ELPA for detection of MDN

Category	Feature	Feature no.
Node characteristics	Characteristics of web links in MDN (proposed)	Feature 3
	Number of domain in LPM (modified)	Feature 2
	Number of node in LPM (modified)	Feature 1
JavaScript obfuscation	Length of longest line in a source code (proposed)	Feature 4
	Ratio of white-space (proposed)	
	Number of lines in the source code (proposed)	

ELPA=Emulation-based Linked Page Map Analysis, MDL=Malware Distribution Network, LPM=Linked Page Map.

Feature 1. The number of nodes in a LPM: The first feature is the number of nodes in a LPM. The attacker makes the MDN by hacking a regular website. Therefore, the MDN has more nodes than a non-MDN. Fig. 3 shows the comparison of the number of nodes in the LPM of the MDN and non-

MDN. As illustrated in the figure, in the case of the non-MDN, most of LPMs have single or double nodes compared to the LPM of the MDN, which has multiple nodes. Thus, this number can be used as a feature of LPM. This feature can be represented as Eq. (2):

$$Feature1 = MAX_{\{j\}},\ j \in LPM_{(i),(j)} \tag{2}$$

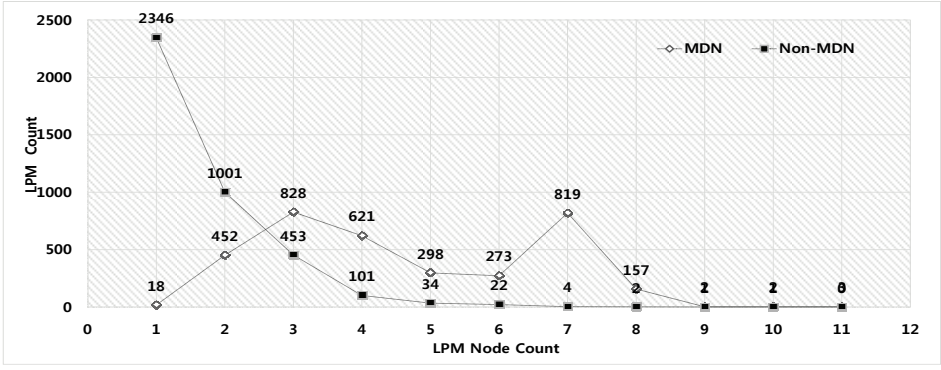

Fig. 3. Linked Page Map (LPM) node count of Malware Distribution Network (MDN) and non-MDN.

Feature 2. Number of domains in a LPM: As previously mentioned, a MDN usually consists of various types of webpages and websites. In addition, the sites that spread the malicious code have a different domain that has been in operation. Because it is created by the attacker. Fig. 4 illustrates the number of different domains used in the MDNs and non-MDN. We can see that the MDN contains multiple domains; whereas, most of the non-MDNs consist of single or two domains. Based on this result, we set the number of domains used in LPM as a feature. Feature 2 can be represented as Eq. (3):

$$Feature2 = n\left(\bigcap LPM_{(i),(j)}\{Domainname\}\right) \tag{3}$$

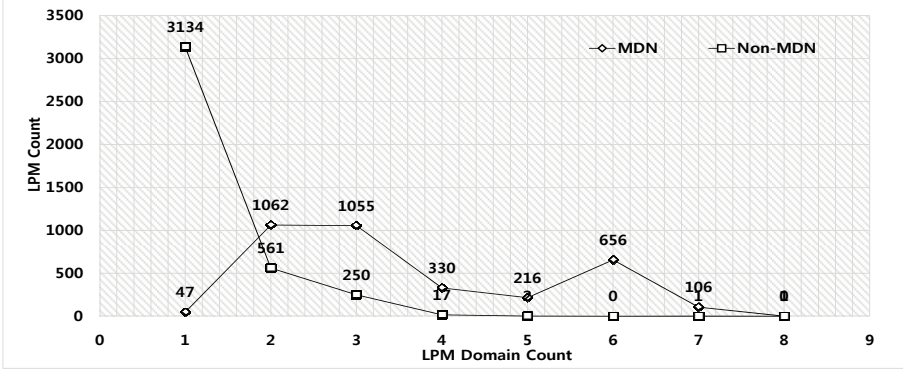

Fig. 4. Number of domains in Malware Distribution Network (MDN) and non-MDN.

Feature 3. Characteristics of a web link in a MDN: In a drive-by download attack, a victim is redirected through a path consisting of a landing site, multiple hopping sites, and an exploiting site. As illustrated in Fig. 5, some hopping sites are shared by multiple MDNs. One may think that the number

of hopping sites and the linking method between the sites can be a feature that represents the MDN. However, since the number of sites and linking method can vary depending on the attackers who construct MDNs, they are not an appropriate feature.

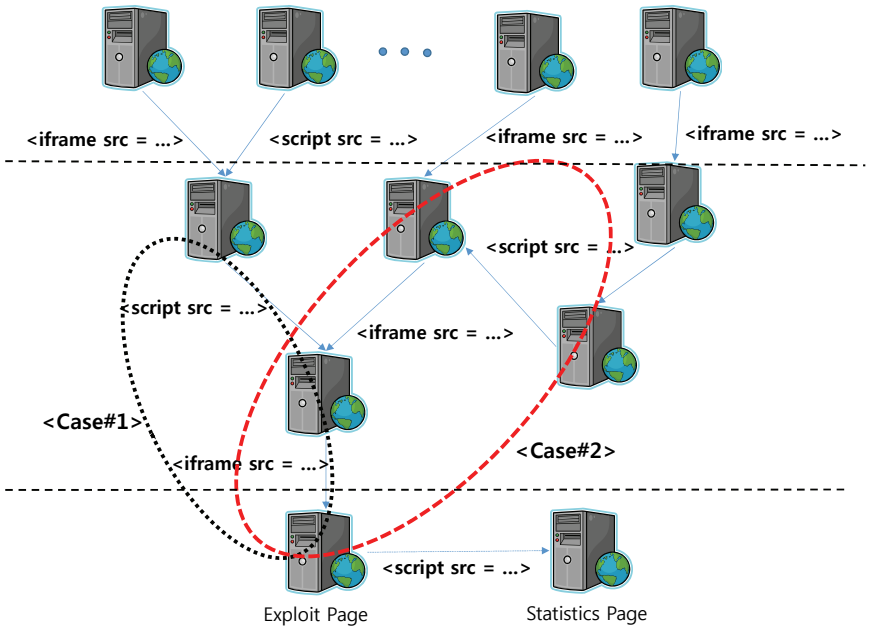

Fig. 5. Characteristic of Linked Page Map (LPM) link.

On the other hand, in our dataset, we found that the file extension of most exploiting pages is HTML, rather than an image and JavaScript extensions, as illustrated in Fig. 6. Related to this, we found in our analysis results that the obfuscated website for malware distribution does not contain parameters. Moreover, the source codes of distribution webpages are mostly obfuscated. Thus, we can characterize a malware distribution page with the existence of parameter in a URL and whether the page is encrypted. Eq. (4) represents Feature 3. We will introduce the feature that represents encryption as Feature 4.

$$Feature3 = n\left(LPM_{(i),(j)} \mid string("?") \notin LPM_{(i),(j)}\{Nodename\} \wedge Encrypt\right) \tag{4}$$

Fig. 6. Characteristic of malware distribution page.

Feature 4. Obfuscation (Encryption): Attackers use various forms of obfuscation techniques in JavaScript to make it difficult to analyze. These techniques utilize the built-in functions in JavaScript, such as unescape(), replace(), parseInt(), split(), charCodeAt(), fromCharCode(), write(), eval(), and sub-string(). However, these functions can be used in benign scripts to protect their source codes. In a normal webpage or JavaScript, only a portion of the contents is obfuscated. In contrast, attackers usually obfuscate the entire contents of the source code to conceal the exploit code. To characterize this, we chose three features of a webpage source code that include the length of the longest line (X), the ratio of white space in the longest line (Z), and the number of lines in the webpage source code (Y). Feature 4 is represented as Eq. (5). In the equation, ML indicates the length of the longest line in the source code, SZ means the number bytes allocated, SSZ means the number of bytes allocated for a white-spaced character, and LCT indicates the number of lines in the source code.

$$Feature4 = \left\{ X,Y,Z \mid X = \frac{SZ\left(ML\left(LPM(i),(j)\right)\right)}{SZ\left(LPM(i),(j)\right)} \times 100, Y = LCT\left(LPM(i),(j)\right), Z = \frac{SSZ\left(ML\left(LPM(i),(j)\right)\right)}{SZ\left(ML\left(LPM(i),(j)\right)\right)} \times 100 \right\} \quad (5)$$

In contrast to the existing features, the proposed four features do not only rely on the contents of the webpage, but on the behavior and structure of the malware distribution process. Thus, the proposed features are more robust against changes in attack methods and strategies. In the following subsection, we introduce the detection algorithm of ELPA, which utilizes the proposed features.

3.3 Algorithm for ELPA

The MDN detection algorithm for ELPA is presented in Table 4. It checks whether the LPM satisfies all four features described in Section 3.2 and decides on its maliciousness. In Step 1, the algorithm initializes the LPM structure and the number of nodes and domains are counted in Step 2 (Features 1 and 2). In Step 3, it examines if the webpage is obfuscated and whether any parameter values exist in the URL. Step 4 finalizes the maliciousness of the LPM based on the analysis result of the previous steps.

Table 4. Algorithm of ELPA

input	$LPM_{(i).(0)}$,$LPM_{(i).(1)}$,…,$LPM_{(i).(m)}$: all LPM
	X : Critical Value of encryption condition #1
	Y_{min} : Minimum critical Value of encryption condition #2
	Y_{max} : Maximum critical Value of encryption condition #2
	Z : Critical Value of encryption condition #3
	$F_{(1)}$: Critical Value of Feature1
	$F_{(2)}$: Critical Value of Feature2
output	Landing page : Intial access web-site
	Hopping page : Auto linked web-site
	Exploit page : Malware Distribution web-site
declare	$Count\_F_{(i)}$: True count of Features, initially 0
	Count_HN : Unique count of $LPM_{(i).(j)}$.Domaintname
	Count_Node : Count of $LPM_{(i).(j)}$
	Value_F3 : Boolean Value of Feature3, initially True

ML : Longest Line, initially 0
SZ(a, b) : All "b" character's size in "a"set, initially 0
LCT() : Line count, initially 0
EP : Boolean value of encryption condition, initially False

algorithm MDN_Decision
begin
STEP 1: //initialization
Landing page = NULL
Hopping page = NULL
Exploit page = NULL
STEP 2: //Feature1, Feature2
loop
While j < m such that $LPM_{(i).(j)}$ **do**
if $LPM_{(i).(j)}$.Domainname not same All Domainname from $LPM_{(i).(0)}$.Domainname to $LPM_{(i).(j-1)}$ **then**
Count_HN = Count_HN +1
endif
save Count_HN
endwhile
save Count_Node = m+1

STEP 3: //Frature3, Feature4
loop
While j < m such that $LPM_{(i).(j)}$ **do**
foreach DContents = Downloaded Contents of $LPM_{(i).(j)}$ **do**
ML = Lognest Line of DContests
if (SZ(ML, *)/SZ(DContents, *))*100 > X
and Y_{min} < LCT(DContents) < Y_{max}
and (SZ(ML, "space")/SZ(ML, *))*100 < Z **then**
EP($LPM_{(i).(j)}$) = True
String("?") ∉ $LPM_{(i).(j)}$.Nodename
Value_F3($LPM_{(i).(j)}$) = True
endif
endfor
endwhile

STEP 4: //Decision MDN
if Count_Node >= $F_{(1)}$ and Count_HN >= $F_{(2)}$ and
∃ {$LPM_{(i).(j)}$|EP($LPM_{(i).(j)}$) = True and Value_F3($LPM_{(i).(j)}$) = True} **then**
Descision LPMc is MDN
return Landing page = $LPM_{(i).(0)}$
return Hopping page = {$LPM_{(i).(j)}$| 1≦ j < m}
return Exploit page = $LPM_{(i).(m)}$
endif

end MDN_Decision

4. Evaluation

4.1 Evaluation Environment

We evaluated the performance of ELPA by comparing it with the URL scanning engine of VirusTotal [17]. Fig. 7 illustrates the evaluation environment. During the two-week evaluation period, the test bed periodically downloaded 508 live websites and ELPA fetched a LPM from the websites using the web browsing emulator [19] and analyzed it. Then, ELPA decided whether the LPM was a MDN by examining it with the features described in Section 3. In addition, we used an open API provided by VirusTotal to submit the URL information to its URL scanning engine and identified the maliciousness of the webpage.

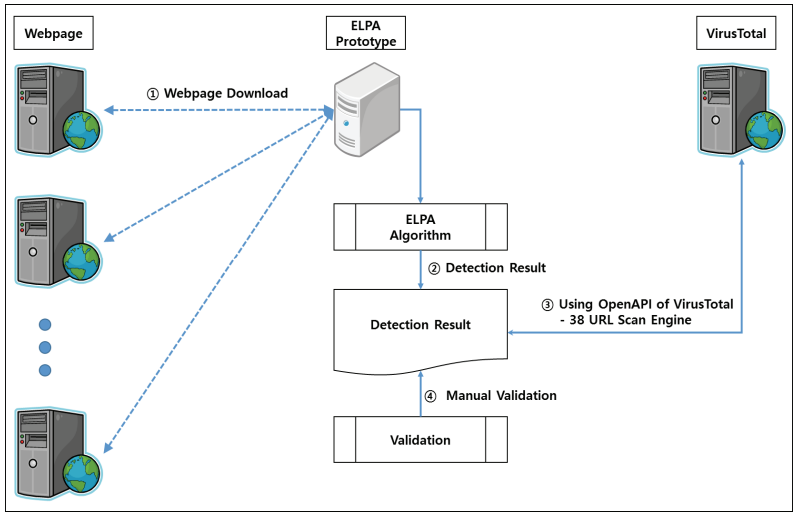

Fig. 7. Evaluation environment. ELPA=Emulation-based Linked Page Map Analysis.

4.2 Website URL Dataset

We selected 508 live websites that have been compromised at least once by an attacker for a drive-by download attack within a year and used them as the seed for our dataset. Since the status of the websites can be changed during the year, we examined the websites using a two-phase evaluation. In the first phase, we input the URLs of websites into VirusTotal and saw the decision of detection engines. If all of the engines made a decision that a website is a clean site, we categorized it as a benign site. Otherwise, we manually examined its maliciousness. Table 5 summarizes how we categorized the websites in our seed dataset.

Table 5. Validation of website dataset

Phase-1 (VirusTotal)	Phase-2 (manual validation)	Maliciousness
Benign	N/A	Benign
Malicious	Malicious	Malicious
	Benign	Benign

4.3 Evaluation Results

During the two-week evaluation, the ELPA periodically visited and checked all 508 websites 42 times. At the beginning of the evaluation, we found that three websites were still compromised and used as a landing site of a MDN. Fig. 8 illustrates the number of MDNs detected during the evaluation period. In the entire period, ELPA found 346 MDNs and some webpages were detected in consecutive rounds (represented as dotted circles in the figure). This is due to the fact that if a compromised website has not been fixed within two rounds, ELPA can detect an MDN rooted from the website in both rounds.

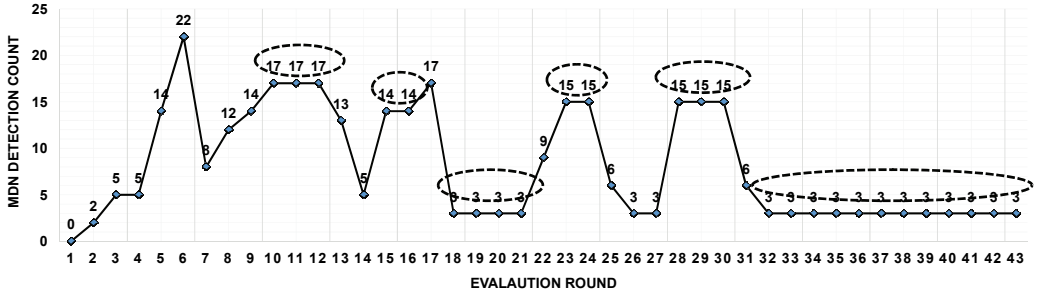

Fig. 8. Number of detected Malware Distribution Network (MDN).

Out of the 346 MDNs in total, there were 17 unique MDNs and malicious (or compromised) webpages that constructed those MDNs. We gave the URLs of the webpages to VirusTotal. Table 6 shows our comparison results of malicious webpage detection. As shown in the table, 75% of malicious webpages detected by ELPA can be detected using existing solutions supported by VirusTotal at most. This means ELPA is a more robust method that advances evasion methods more than all of the other existing solutions listed in the table.

Table 6. Summary of detection result

Detection engine	No. of hopping & exploit page	Comparison with ELPA (%)
ELPA	37	-
Emsisoft	28	75.68
BitDefender	26	70.27
malwares.com URL checker	25	67.57
Fortinet	16	43.24
CLEAN MX	12	32.43
Google Safebrowsing	12	32.43
Sophos	12	32.43
Avira	10	27.03
SCUMWARE.org	8	21.62
Antiy-AVL	7	18.92
TrendMicro	6	16.22
Webutation	5	13.51
Yandex Safebrowsing	3	8.11
ADMINUSLabs	1	2.70
AutoShun	1	2.70
Dr.Web	1	2.70
Websense ThreatSeeker	1	2.70
ESET	0	0
AutoShun	0	0
WOT	0	0

5. Conclusion

Even though numerous methods have been proposed to detect drive-by download attacks, existing methods have limitations when it comes to analyzing various types of JavaScript obfuscation, hiding, and evasion techniques. In this paper, we have proposed the ELPA method, which detects and analyzes MDNs with a feature-based analysis of web links. ELPA first extracts the LPM from the web link structure, then determines its maliciousness based on the proposed features. Our performance evaluation shows that the detection performance of ELPA outperforms existing solutions provided by VirusTotal.

In our future work, we will improve ELPA to precisely extract LPMs from a chain of webpages that use more advanced evasion techniques. Moreover, we plan to resolve some limitations of the current ELPA method, such as detecting the case where a landing page is the only node in a MDN.

References

[1] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu, "The ghost in the browser analysis of web-based malware," in *Proceedings of the 1st Conference on First Workshop on Hot Topics in Understanding Botnets*, Cambridge, MA, 2007, pp. 1-9.

[2] European Union Agency for Network and Information Security, "ENISA Threat Landscape 2012", Jan. 2013; https://www.enisa.europa.eu/activities/risk-management/evolving-threat-environment/enisa-threat-landscape/ENISA_Threat_Landscape.

[3] F. Y. Rashid, "Department of labor website hacked to distribute malware," May 2013; http://www.securityweek.com/department-labor-website-hacked-distribute-malware.

[4] J. Pepitone, "NBC hack infects visitors in 'drive by' cyberattack," Feb. 2013; http://money.cnn.com/2013/02/22/technology/security/nbc-com-hacked-malware/.

[5] E. Protalinski, "A first: Hacked sites with Android drive-by download malware," May 2012; http://www.zdnet.com/article/a-first-hacked-sites-with-android-drive-by-download-malware/.

[6] HAURI, "Malware analysis report of NateOn hacking" Aug. 2012; http://eyesray.tistory.com/attachment/cfile10.uf@1615E0564E3DDD17213F95.pdf

[7] G. Cluley, "DarkSeoul: SophosLabs identifies malware used in South Korean internet attack," Mar. 2013; http://nakedsecu rity.sophos.com/2013/03/20/south-korea-cyber-attack.

[8] ASEC, "Malware analysis report using 6.25 DDoS attack," Jun. 2013; http://asec.ahnlab.com/949.

[9] K. Z. Chen, G. Gu, J. Zhuge, J. Nazario, and X. Han, "WebPatrol: automated collection and replay of web-based malware scenarios," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, Hong Kong, 2011, pp. 186-195.

[10] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious URLs: an application of large-scale online learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, Montreal, Canada, 2009, pp. 681-688.

[11] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious URLs," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Paris, France, 2009, pp. 1245-1254.

[12] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose, "All your iFRAMEs point to us," in *Proceedings of 17th USENIX Security Symposium*, San Jose, CA, 2008, pp. 1-16.

[13] A. Moshchuk, T. Bragin, D. Deville, S. D. Gribble, and H. M. Levy, "SpyProxy: execution-based detection of malicious web content," in *Proceedings of 16th USENIX Security Symposium*, Boston, MA, 2007, pp. 1-16.

[14] M. Cova, C. Kruegel, and G. Vigna, "Detection and analysis of drive-by-download attacks and malicious JavaScript code," in *Proceedings of the 19th International Conference on World Wide Web*, Raleigh, NC, 2010, pp. 281-290.

[15] B. Genge and C. Enachescu, "ShoVAT: Shodan-based vulnerability assessment tool for Internet-facing services," *Security and Communication Networks,* 2015. http://dx.doi.org/10.1002/sec.1262.

[16] B. Genge and C. Enachescu, "Non-intrusive historical assessment of internet-facing services in the internet of things," *MACRo*, vo. 1, no. 1, pp. 25-36, 2015.

[17] VirusTotal, [Online]. Available: https://www.virustotal.com/.

[18] J. H. Kim, "Understanding of Javascript obfuscation," May 2008; http://image.ahnlab.com/file_upload/tech/javascript.pdf.

[19] S. Y. Choi, I. S. Kang, D. H. Kim, B. N. Noh, and Y. M. Kim, "Multi-level emulation for malware distribution networks analysis," *Journal of the Korea Institute of Information Security & Cryptology*, vol. 23, no. 6, pp. 1121-1129, 2013.

[20] SpiderMonkey [Online]. Available: https://developer.mozilla.org/en-US/docs/Mozilla/Projects/SpiderMonkey.

Sang-Yong Choi http://orcid.org/0000-0001-5152-3897

He received his B.S. degree in Mathematics and M.S. degree in Computer Science, both from Hannam University in 2000 and 2003, and Ph.D. degree in Interdisciplinary of Information Security from Chonnam National University in 2014, Korea. He is a principal researcher at the Cyber Security Research Center in Korea Advanced Institute of Science and Technology (KAIST). His research interests are in web security, network security and privacy.

Daehyeok Kim http://orcid.org/0000-0002-7439-1783

He received his B.S. degree in Computer Science and Engineering and M.S. degree in IT Convergence Engineering, both from Pohang University of Science and Technology (POSTECH), Korea. He is a senior researcher at the School of Computing in Korea Advanced Institute of Science and Technology (KAIST), Korea. His research interests are in distributed systems, networking, and their security and privacy.

Yong-Min Kim http://orcid.org/0000-0002-5066-3908

He received his Ph.D. in Dept. of Computer Science and Statics in Chonnam National University, Korea. He is an associate professor at Dept. of Electronic Commerce, Chonnam National University, Yeosu, Korea. His research interests are in security and privacy, system and network security, application security as electronic commerce.