Malicious script distribution pattern detection technique for image search websites

Yong-joon Lee[a] **and Won-shik Na**[b,*]
[a]Far East University, Department of Hacking Security, Gamgok-myeon, Republic of Korea
[b]Namseoul University, Department of Computer Science, Cheonansi, Republic of Korea

Abstract. Recently, the number of cases of distributing malicious codes by exploiting homepages that provide an image search continues to increase, and malicious codes distributed through homepages are causing personal information infringement accidents and DDoS attacks. Due to the malware spread through web pages, privacy theft and infringement are getting serious and DoS attacks happen frequently. Distribution patterns of hidden malicious codes on the image search website were collected, and patterns of collected malicious codes and malicious scripts were analyzed. We have analyzed the malicious samples and derived some additional distribution patterns of web-based malware. Similar patterns are grouped together and a representative feature is then extracted from each group. Each category of the malicious samples contains malicious script codes and their variants. We have implemented a system to automatically detect malicious web sites using the malicious script patterns. The proposed malicious script pattern is expected to be available for the zero-day attacks. © *The Authors. Published by SPIE under a Creative Commons Attribution 4.0 International License. Distribution or reproduction of this work in whole or in part requires full attribution of the original publication, including its DOI.* [DOI: 10.1117/1.JEI.31.3.033046]

Keywords: image search websites; malicious code detection system; malicious script pattern; web security; hopping site; exploit site.

Paper 220111G received Feb. 4, 2022; accepted for publication Jun. 15, 2022; published online Jun. 28, 2022.

1 Introduction

The personal information breach and DDos attack, stealing personal information after infecting a user's PC via malicious code, are recently on the rise. Among various routes of the infection, distributing malicious codes after exploiting the vulnerabilities of websites and user PCs is constantly growing.[1] There are two types of infectious websites containing malicious code: one is a distribution site that conceals malicious code in it, and the other is an intermediate site that automatically connects a PC to a distribution site through an address implanted in the code. Hackers open up a distribution site containing malicious code, hack an intermediate site, and insert the URL of the distribution site. As such, any visitor to an intermediate site would be unwittingly induced to a distribution site and have his/her PC infected with malicious code. Moreover, high-traffic websites, such as portals, blogs, or bulletin boards, become the target of hackers. Thus, detecting malicious code is getting more difficult as hackers obfuscate such code before implanting on the target websites.[2] The methods of detecting hidden malicious codes on a website can be categorized into two. The first one is the signature-based detection method, which intends to check where malicious code is inserted in the source code of a website.[3] It shows a fast-detection speed but lower detection performance in the face of a zero-day attack. The second one is the behavior-based detection method, which is designed to detect malicious code by accessing a website and tracking status changes, such as the modulation of a visitor's PC files or the execution of malicious code for detection.[4] It manifests slow detection speed but shows high detection performance for a zero-day attack. This paper proposes a new detection method, which is distinguished from the existing methods, to check the obfuscation of malicious web page scripts for a pattern of detection.[5] It is designed to extract the patterns of malicious distribution by analyzing malicious scripts distributed via web pages.[6] As the extracted patterns

*Address all correspondence to Won-shik Na, winner@nsu.ac.kr are registered and checked in detection rules, the method maintains the speed of the signature-based detection. It can also detect a zero-day attack.[7] Research on pattern and technology trends of distribution of malicious codes. Through this process, it presents trends of changes in patterns of distribution of malicious code and predicted trends in the future through analysis of patterns of distribution of new malicious codes. Based on collected distribution patterns, it analyzes vulnerability types and patterns and develops detection algorithms and similarity comparison algorithms by distribution patterns of malicious codes.[8] It is going to develop automated modules that analyze patterns of distribution of malicious codes by applying developed algorithms.[9] In this paper, we compared three recent methods for detecting malicious codes on image providing websites. We compared the signature-based detection method, the behavior-based detection method, and the script pattern-based detection method. The purpose of this paper is to efficiently detect malicious codes that target image providing sites.[10] The limitation of this paper is that it is difficult to collect various malicious code scripts.[11] Moreover, this paper consists of five sections. Section 2 describes the security vulnerabilities of web application and the distribution paths of malicious code in the existing study and provides the analysis and classification of the study on website detection methods. Meanwhile, Section 3 demonstrates the analysis of distribution patterns in malicious scripts and describes the website detection method. Section 4 explains the experiment conducted based on the proposed method and its results. Lastly, Section 5 shows the conclusion of this paper.

2 Existing Study

2.1 *Security Vulnerabilities of Web Application*

Hackers exploit the vulnerabilities of a web application to conceal malicious code within web pages used as an intermediate site for distribution and spreading of malicious code.[12] Open Web Application Security Project (OWASP) defines the security vulnerabilities of a web application targeted by hackers, as indicated in Table 1.

2.2 *Paths of Malicious Code Distribution*

There are two main paths where PCs of Internet users get infected with malicious code: one is via a distribution site hiding malicious code in it, and the other is via an intermediate site concealing

Table 1 Web application security vulnerability and response method.

Security vulnerability	Response method
Cross-site scripting	Standard validation of input values, encrypting output values
Injection vulnerability	Parameterized query language, object relationship mapping
Remote file execution	Network independent configuration, design phase verification technique
Unstable direct object reference	Indirect reference map, allowing direct reference after authorization
Cross-site request modulation	Separate authentication mechanism, a POST request method
Information leakage and improper error handling	Restricting disclosure of error information, disabling error information disclosure function
Vulnerable authentication and session management	Single authentication mechanism, separate session management mechanism
Unstable encryption storage	Proven cryptographic algorithm implementation, secure encryption key management
Unstable communication	SSL
Failure to restrict URL access	Implementing access metric

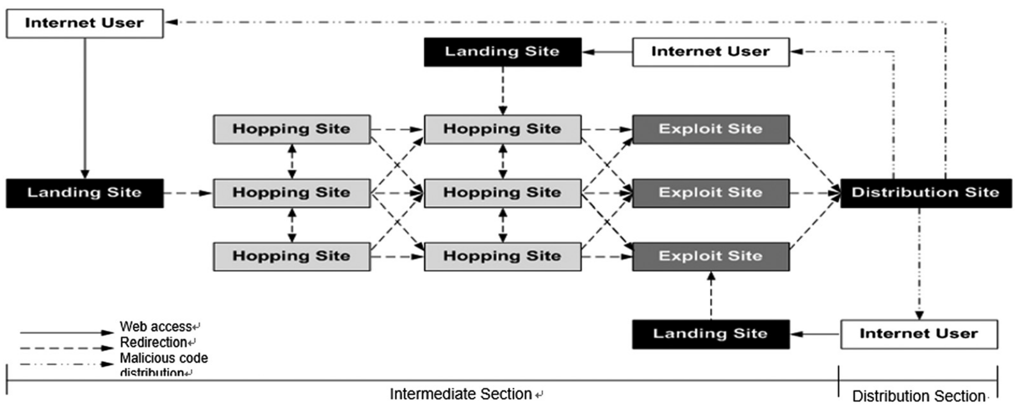

Fig. 1 Malicious code distribution routes.

executable code redirecting a page automatically to a distribution site. Figure 1 shows an intermediate section and a distribution section used in the distribution of malicious code on web pages.[13]

- Initial intermediate site (landing site): An initial intermediate site is the first web page that induces Internet users to a malicious distribution site and serves as a gateway as well. In general, high-traffic Internet portals and blogs providing information on social issues can be exploited as an initial intermediate site. The users are redirected to a malicious site by implanted iframe codes connecting to an assigned web page or links of key words for social issues.
- In-between intermediate site (hopping site): An in-between intermediate site serves simply as a medium to send users from the initial intermediate site or a previous intermediate site to the next intermediate site. Therefore, poorly managed web pages, instead of high-traffic web pages dealing with social issues, are more likely to be exploited as intermediate sites. Moreover, codes connecting a page to an assigned web page, such as iframe, are usually inserted.
- Final intermediate site (exploit site): Just like other in-between intermediate sites, poorly managed websites are used as a final intermediate site. Their role is simply more than a medium as they exploit malicious script attacking the vulnerabilities of a PC, and an application is implanted. In this paper, we analyze the distribution patterns of exploit malicious scripts and use them in detecting web pages that distribute malicious code.
- Distribution site: A malicious distribution site is a web page infecting user PCs with malicious code, and it is either a website run directly by a hacker or a hacked website being exploited. They do not have any malicious code implanted to get involved in distribution, but rather automatically installs another type of malicious code without the consent of users during access.[14]

2.3 *Malicious Code Detection System*

Google Safe Browsing, MS HoneyMonkey, and UW Spycrawler are the representative malicious code detection systems overseas. They are intended to detect a malicious distribution site, a malicious zero-day attack, a spyware website, and others.[15]

- Google Safe Browsing: Google Safe Browsing provides the security of visiting websites by sending a blacklist of collected distribution sites every 30 min to the user's web browser. If a user accesses a malicious distribution site, it would display an alert to the user and share the issues of the website.[16] Figure 2 shows the process of Google's detection and analysis of malicious distribution sites. The automated detection and analysis consist of three steps: a suspected URL identification step, a suspected URL verification step, and a risk rating-based URL collection step. First, the system identifies URLs distributing malicious code by crawling a website. Second, it uses a virtual machine's Internet explorer and

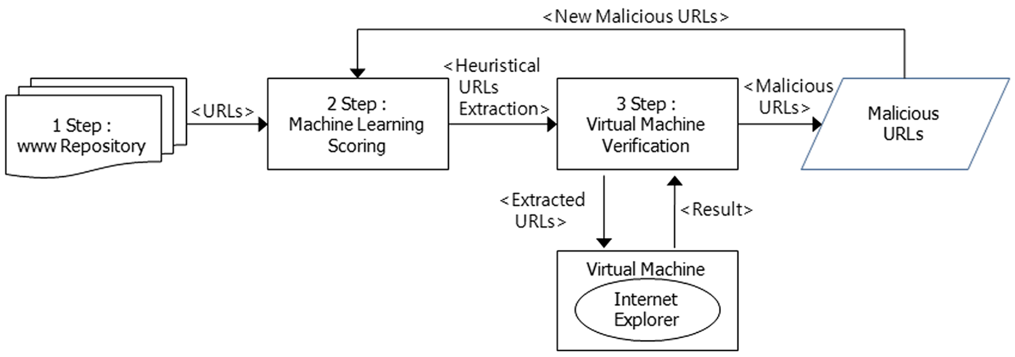

Fig. 2 Google's Safe Browsing malicious code detection system.

records the status changes of the suspected URL's HTTP and the machine to verify the cause of the vulnerabilities of the web browser. Third and last, it registers the URL to the blacklist if it is malicious. Google Safe Browsing is a blocklist service provided by Google that provides lists of URLs for web resources that contain malware or phishing content. The Google Chrome, Safari, Firefox, Vivaldi, and GNOME Web browsers use the lists from the Google Safe Browsing service for checking pages against potential threats.[17] Google also provides a public API for the service. Google also provides information to Internet service providers, by sending e-mail alerts to autonomous system operators regarding threats hosted on their networks.[18] According to Google, as of September 2017, over 3 billion Internet devices are protected by this service.[19]

- MS's HoneyMonkey: MS HoneyMonkey detects attacks toward the client in Window and Internet explorer environments by monitoring files, registries, and processes of the virtual machines accessed to malicious distribution sites. To identify a zero-day attack, it confirms a downloaded malicious code by crawling the distribution site via a low security-patch web browser and then verifying whether the attack can be defended with available security patches by increasing the level of a security patch of the web browser one by one.[20] Then, the attack would be categorized as a high-risk zero-day attack if the security patches cannot block it. As shown in Fig. 3, HoneyMonkey analyzes a web browser's process and network packets using Browser Helper Object in a virtual environment. It monitors changes in executable files, processes, registries, and reassigned URLs within the web browser's sandbox to identify a distribution site. Once a distribution site is confirmed, HoneyMonkey takes three-step actions to check on a zero-day attack. First, it identifies a malicious distribution site by accessing it via a web browser without any security patch installed. Second, it defines the risk rating after analyzing the number and relevance of redirect-URL on the identified distribution site. Third and last, it checks the existence of a zero-day attack by accessing the malicious distribution site via a web browser with a security patch installed.[21] HoneyMonkey, short for Strider HoneyMonkey Exploit Detection System, is a Microsoft Research honeypot. The implementation uses a network of computers to crawl the World Wide Web searching for websites that use browser exploits to install malware on the HoneyMonkey computer.[22]

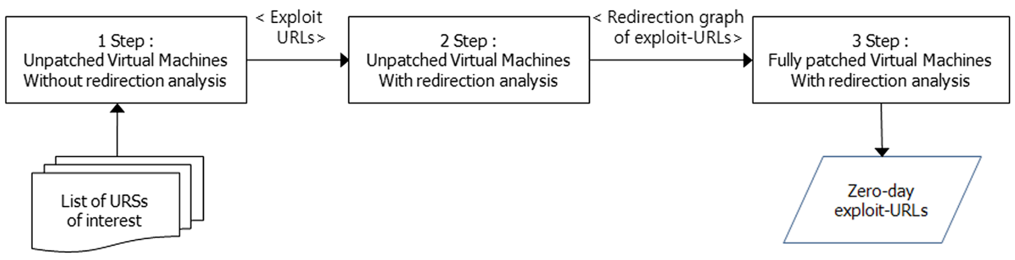
1 Step : Unpatched Virtual Machines Without redirection analysis
< Exploit URLs>
2 Step : Unpatched Virtual Machines With redirection analysis
< Redirection graph of exploit-URLs>
3 Step : Fully patched Virtual Machines With redirection analysis
List of URSs of interest
Zero-day exploit-URLs

Fig. 3 MS's HoneyMonkey malicious detection system.

A snapshot of the memory, executables, and registry of the honeypot computer is recorded before crawling a site.[23] After visiting the site, the state of memory, executables, and registry is recorded and compared with the previous snapshot.[24] The changes are analyzed to determine if the visited site installed any malware onto the client honeypot computer. HoneyMonkey is based on the honeypot concept, with the difference that it actively seeks websites that try to exploit it.[25]

- The University of Washington Spycrawler (UW Spycrawler): The University of Washington proposed three-step detection methods to conduct research on spyware via website crawling. First, the system detects all executable files of a website with the assumption that all of them are likely to be malicious. Second, it performs a behavior-based detection by downloading and executing all the executable files on the web using a virtual machine. Third and last, it analyzes logs to detect the malicious behaviors of spyware, such as Piggy-back and Drive-by download. The University of Washington performed the research on Spycrawler to detect website links where spyware is executed.[26] The Spycrawler developed at the University of Washington is yet another browser-based (Mozilla) high interaction client honeypot developed by Moshchuk et al. in 2005. This client honeypot is not available for download. The Spycrawler is state based and detects attacks on clients by monitoring files, processes, registry, and browser crashes. Spycrawlers detection mechanism is event based. Further, it increases the passage of time of the virtual machine the Spycrawler is operating in to overcome (or rather reduce the impact of) time bombs.[19]

Table 2 shows the comparison of malicious code detection methods between Google, MS, and University of Washington. It indicates that behavior-based detection has low detection performance.

3 Proposed Method of Malicious Script Pattern Analysis

If a website is hacked and the update file is forged, all user PCs that download the file will be infected with malware. For this reason, real-time checking is essential for forgery detection. As shown in Fig. 4, the website sensor network detects update file forgery in the following manner: The website operator registers the update file on the inspection server before distributing it to the users. It is important for the website operator to register the update file and then distribute it to the users. Forgery detection of update files is performed in real time by comparing the hash value of the update file on the website with the registered hash value.

Table 2 Malicious code detection method comparison.

Function	Safe-browsing	HoneyMonkey	Spycrawler
Detection goal	Provides website security by securing malicious distribution sites	Provides a security patch to detect zero-day attacks	Detects spyware
Detection method	Signature-based	Behavior-based	Behavior-based
Detection stages	• Identifies a malicious code distribution site	• Identifies a malicious code distribution site	• Identifies an execution file
	• Analyzes behaviors	• Analyzes behaviors	• Analyzes activities
	• Verifies risk ratings	• Verifies a zero-day attack	• Verifies advertisement websites
Detection performance	Has high detection speed	Has low detection speed	• Has low detection speed
Zero-day detection	Has low detection speed	Has high detection speed	• Has high detection speed

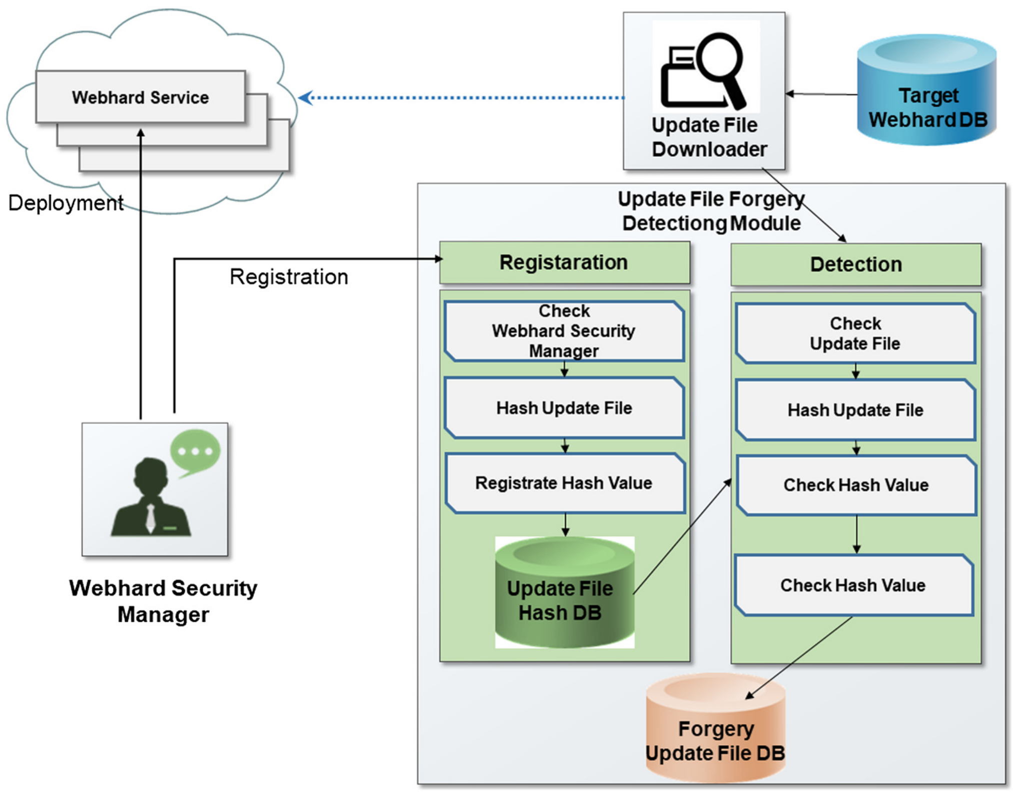

Fig. 4 Architecture to check for update file forgery.

In addition to the existing signature-based analysis and behavior-based analysis, this paper proposes a pattern analysis method of a malicious script to verify malicious distribution sites, which can provide a good detection speed and scalability to the zero-day attack detection.

3.1 *Analyzing Malicious Script Distribution Patterns*

We analyzed malicious script distribution patterns by crawling 500 websites locally and abroad, confirmed as malicious distribution sites, to provide the analysis of malicious script patterns. As a result, 95% of them use a form of web scripts, and the remaining 5% have scripts inserted in multimedia files. The analyzed patterns of distribution are listed in Table 3.

In addition to the existing signature-based analysis and behavior-based analysis, this paper proposes a pattern analysis method of a malicious script to verify malicious distribution sites, which can provide a good detection speed and scalability to the zero-day attack detection.

Table 3 Analysis results on the distribution pattern of malicious scripts.

Analysis item	Use	Not used	Total
Script	470 URLs (94%)	30 URLs (6%)	500 URLs (100%)
Code obfuscation	315 URLs (63%)	185 URLs (37%)	500 URLs (100%)
document.write()	240 URLs (48%)	235 URLs (52%)	500 URLs (100%)
External link	165 URLs (33%)	335 URLs (67%)	500 URLs (100%)
URL encoding	70 URLs (14%)	430 URLs (86%)	500 URLs (100%)

Table 4 Malicious script detection algorithm.

	Malicious script distribution pattern	Detection algorithm
1	User-defined string processing function	function
2	Large-scale string processing function	unescape, replace, split, fromCharCode
3	Simple string iteration	+\“,&\”,+\‘,&\’,\“+,\”&,\‘+,\’&
4	Eval function	eval
5	Large-scale special character	#, $, %, ^, &
6	us-ascii, jscript.encode encoding function	us-ascii, jscript.encode
7	Web pages containing multimedia	jpg, gif, swf, wav
8	Web pages containing execution files	exe
9	Shell script	Shell.Run, ShellExecute
10	img tag abnormal width and height values	width < 4, height < 4

- Script: Scripts were generally used to conceal malicious code. Based on our analysis, 94% of the websites exploited scripts. For the type of scripts, 49% were in javascript, followed by script by 20%, and vbscript by 17%.
- Obfuscation code: Malicious script code is usually obfuscated or modified to be hidden because it can be detected as a signature-based detection method. Based on our analysis, 63% of the scripts were obfuscated.
- document.write() function : It is commonly reported that the document.write() function is used in malicious scripts because it can constitute internal contents when a web browser shows a web page to a user. Thus, it is possible to separate or conceal malicious code in the function and then combine it later when a web page is displayed. Based on our analysis, 48% of the scripts used the document.write() function.
- External link: External links play the most critical role as they have users to access a distribution site from an intermediate site. Based on our analysis, 33% of concealed malicious codes are in external links.
- URL encoding: URL encoding was used to convert distribution URLs to an unrecognizable form to avoid detection. It accounts for 14% in our analysis.

Table 4 shows 10 detection algorithms used in the detection of malicious script patterns. The first and second are the algorithms detecting the encoding and obfuscation of large-scale character strings. The third pattern is used in obfuscating malicious script when a web page source code is dissembled and reassembled. The fourth pattern of eval function is used in obfuscation to change character strings expressed in formula into numbers. The fifth pattern is derived because special characters and symbols are frequently used within script tags in case the source is encoded or obfuscated. The encoding method of us-ascii and jscript.encode in the sixth pattern is added because it is not used in normal web pages and is frequently used in a malicious script. The seventh pattern is the case where a file is called in for the distribution of malicious code as if a multimedia file is called in. The eighth and ninth patterns are to detect the cases where an exe executable file and a web shell script are used in a malicious exploit file. The 10th pattern is to detect the case where a malicious distribution file is secretly called in using the img tag, which shows an image file.

3.2 *Proposed Detection Method Based on Malicious Script Pattern*

The malicious script distribution patterns proposed in this paper can also be detected on normal web pages. Therefore, it is hard to say a web page is exploited as an intermediate/distribution site even if a pattern is detected. However, the web pages showing such patterns can be categorized as

Table 5 Calculation of importance.

		Detection		
	Malicious script distribution pattern	False-positive detection rate	Number of detected cases	Importance
1	User-defined string processing function	27.9%	27,900	72.1
2	Large-scale string processing function	3.3%	3300	96.7
3	Simple string iteration	3.7%	3700	96.3
4	Eval function	10.4%	10,400	89.6
5	Large-scale special character	21.0%	21,000	79.0
6	us-ascii, jscript.encode encoding function	0.1%	100	0
7	Web pages containing multimedia	0%	0	100
8	Web pages containing execution files	1.5%	1500	98.5
9	Shell script	0%	0	100
10	img tag abnormal width and height values	19.4%	19,400	80.6

suspected web pages. Moreover, the importance would be used to improve certainty. In this proposal, we calculated the importance based on the frequency of malicious script patterns used on normal web pages to provide the level of suspicion quantitatively.

Step 1: We used 1000 normal web pages for the detection algorithms to calculate the importance. If the detection rates are also high on normal web pages, the importance would be lower in calculation to improve the accuracy of detection:

$$\text{Importance} = 1 - \frac{\text{number of detected web pages}}{\text{number of total white list}}. \tag{1}$$

Step 2: We can detect many types of malicious distribution patterns from most of the malicious web pages. The final "risk score" of a web page is calculated based on the detection result value and the importance of malicious script distribution algorithms is the number of malicious script distribution pattern algorithms used, is an index number of algorithm, and indicates importance and detection result value, respectively:

$$\text{Score} = \sum_{1 \le K \le n}^{\max} (i_k \times v_k). \tag{2}$$

Table 5 shows the importance calculated after measuring the false positive detection rate of each malicious script distribution algorithm based on a roughly 100K white list provided by the Korea Internet and Security Agency. The first, fifth, and seventh algorithms show low importance because of the high false-positive detection rate. The low importance indicates that the algorithm using the relevant distribution patterns are actively used in normal web pages as well. The proposed method is effective because it detects with a script pattern, the algorithm complexity is low, and the load on memory is small.

3.3 *Cyber Training Education*

Cyber training education is possible through detection through malicious script patterns presented in this paper. Attackers can provide training for hacking through malicious script patterns. On the contrary, defenders can train to detect hacking attacks using malicious scripts. Hacking attacks on mutual websites and battles for hacking detection are possible. These exercises allow high levels of cyber training.

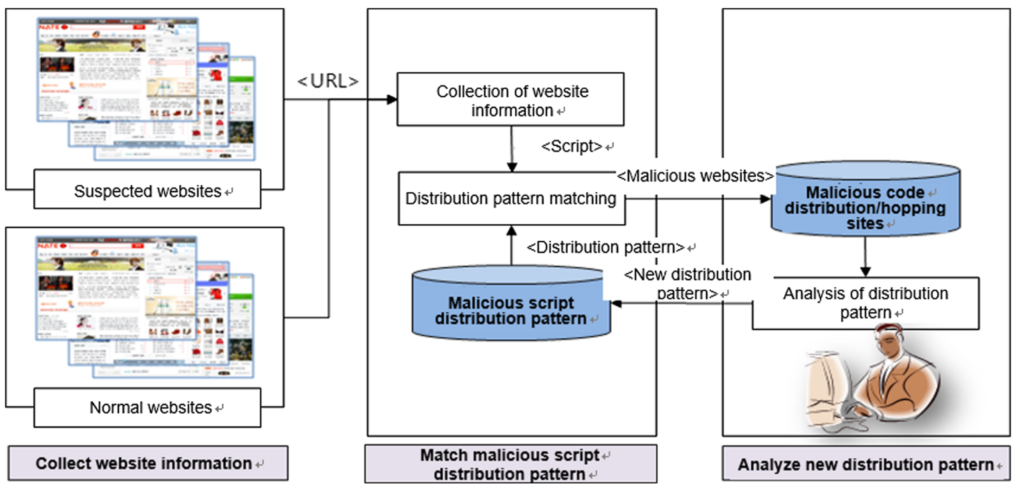
Suspected websites
Normal websites
Collect website information
<URL>
Collection of website information
<Script>
Distribution pattern matching
<Distribution pattern>
Malicious script distribution pattern
Match malicious script distribution pattern
<Malicious websites>
Malicious code distribution/hopping sites
<New distribution pattern>
Analysis of distribution pattern
Analyze new distribution pattern

Fig. 5 Detection system based on a malicious script distribution pattern.

4 Experiment

As shown in Fig. 5, the proposed detection method based on a malicious script distribution pattern is intended to collect information from websites, match distribution patterns, and detect the malicious distribution and intermediate sites.

It is designed to respond to a zero-day attack by constantly adding malicious distribution patterns through new analysis on distribution patterns. Static analysis of the detection system involves collecting a website to be checked by using a crawler, which is a collection tool, and checking the website source code for malicious scripts using anomaly detection patterns. The anomaly detection patterns check the website sources for two types of anomalies, which are website links for spreading malware and malicious scripts for attacking a PC's vulnerabilities. For this reason, it is important to collect and analyze as many cases of malware as possible and register them as detection patterns to detect them when they are concealed in the detection system.

The experiment on the proposed detection method was conducted using algorithms extracted from 500 samples of malicious distribution and hopping sites revealed by Google and MS. The

Table 6 Malicious script detection algorithm.

		Detection		Nondetection		
	Malicious script distribution pattern	Detection rate	Number of detected cases	Non-detection rate	Number of undetected cases	Weight
1	User-defined string processing function	88.8%	178	11.2%	22	0.03
2	Large-scale string processing function	89.4%	161	10.6%	19	0.12
3	Simple string iteration	88.3%	132	11.7%	58	0.12
4	Eval function	100.0%	100	0.0%	0	0.10
5	Large-scale special character	98.0%	196	2.0%	4	0.03
6	us-ascii, jscript.encode encoding function	100.0%	20	0.0%	0	0.15
7	Web pages containing multimedia	97.6%	49	2.4%	1	0.15
8	Web pages containing execution files	98.7%	79	1.3%	1	0.10
9	Shell script	100.0%	50	0.0%	0	0.15
10	img tag abnormal width and height values	100.0%	5	0.0%	0	0.05

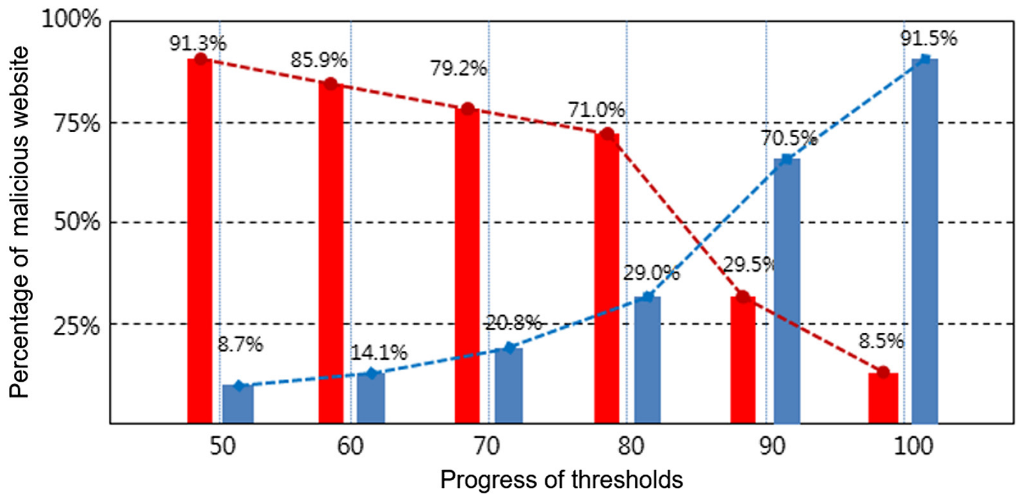

Fig. 6 Percentage of suspected web pages depending on changes in thresholds.

weight was added to the algorithms based on importance. If the calculated risk score reaches 60 and above in the experiment, it is considered detected. The results based on the malicious distribution pattern algorithm are shown in Table 6. Algorithms show high detection rates except for the first, second, and third ones in Table 6. These three algorithms show a relatively high non-detection rate because of the limitations in developing detection algorithms; there are various ways of dissembling and reassembling character strings.

Figure 6 shows changes in the percentages of suspected web pages depending on thresholds. The higher the threshold is set, the lower the risk rate of web pages is. In particular, the percentage sharply drops around 86 points. Thus, it is possible to analyze websites based on risk importance using the proposed malicious script distribution patterns.

5 Conclusion

Although the distribution of malicious code exploiting the vulnerabilities of websites and web browsers is recently increasing, the existing detection methods of malicious code do not consider the characteristics of the web. There are two existing methods of malicious code detection: a signature-based detection to identify malicious code and a behavior-based detection to track the status changes on a website. The signature-based detection provides a high detection speed but is not effective in detecting a zero-day attack, whereas the behavior-based detection is effective for a zero-day attack but has slow detection speed. Therefore, a detection method in the consideration of the characteristics of the web is required to effectively detect attacks on websites. This paper proposed a method to detect the distribution and intermediate sites based on malicious script distribution patterns. We extracted distribution patterns through the analysis and classification of common characteristics after analyzing the scripts of malicious distribution sites. Whereas the existing detection method focuses on the execution of malicious code, the proposed distribution pattern detection method uses the analysis of client scripts to consider the characteristics of the web to improve the speed and capabilities of detection. As the number of incidents where malicious distribution sites exploit the vulnerabilities of smart devices is growing, research on malicious script distribution patterns for the mobile web is necessary in the future. The purpose of this study is to introduce a system for detecting malware distributed through websites. It is expected that this system will detect malware quickly and scientifically and grasp the hidden purpose of the attackers. The advantage of this method is that it can detect patterns quickly, but the downside is that it cannot detect new, unregistered malware.

Acknowledgments

Funding for this paper was provided by Namseoul University.

References

1. B. A. Khalaf et al., "Comprehensive review of artificial intelligence and statistical approaches in distributed denial of service attack and defense methods," *IEEE Access*, **7**, 51691–51713 (2019).
2. Y. Wang et al., "Detecting stealth software with strider GhostBuster," in *Proc. DSN*, pp. 368–377 (2005).
3. S. Singhal, U. Chawla, and R. Shorey, "Machine learning and concept drift based approach for malicious website detection," in *Int. Conf. Commun. Syst. & Netw. (COMSNETS)*, Bengaluru, India, pp. 582–585 (2020).
4. K. Rieck et al., "Learning and classification of malware behavior," *Lect. Notes Comput. Sci.* **5137**, 108–125 (2008).
5. H. Mishra, R. K. Karsh, and K. Pavani, "Anomaly-based detection of system-level threats and statistical analysis," in *Smart Comput. Paradigms: New Progr. and Chall.*, pp. 271–279 (2019).
6. J. Wang, A. Ghosh, and Y. Huang, "Web canary: a virtualized web browser to support large-scale silent collaboration in detecting malicious web sites," in *Proc. CollaborateCom*, pp. 24–33 (2008).
7. P. Likarish, E. Jung, and I. Jo, "Obfuscated malicious javascript detection using classification techniques," in *Proc. MALWARE*, pp. 47–54 (2009).
8. R. Panigrahi et al., "A consolidated decision tree-based intrusion detection system for binary and multiclass imbalanced datasets," *Mathematics* **9**(7), 751 (2021).
9. S. A. Mostafa et al., "Formulating layered adjustable autonomy for unmanned aerial vehicles," *Int. J. Intell. Comput. Cybern.* **10**, 430–450 (2017).
10. C. L. Chowdhary et al., "Analytical study of hybrid techniques for image encryption and decryption," *Sensors* **20**(18), 5162 (2020).
11. N. Khan, J. Abdullah, and A. S. Khan, "Defending malicious script attacks using machine learning classifiers," *Wireless Commun. Mobile Comput.* **2017**, 9 (2017).
12. X. Lu et al., "A universal malicious documents static detection framework based on feature generalization," *Appl. Sci.* **11**(24), 12134 (2021).
13. Y. Hou et al., "Malicious web content detection by machine learning," *Expert Syst. Appl.* **37**(1), 55–60 (2010).
14. D. Liu and J. H. Lee, "CNN based malicious website detection by invalidating multiple web spams," *IEEE Access* **8**, 97258–97266 (2020).
15. M. Cova, C. Krügel, and G. Vigna, "Detection and analysis of drive-by-download attacks and malicious javascript code," in *Proc. WWW*, pp. 281–290 (2010).
16. M. So-Yeon et al., "Design of comprehensive security vulnerability analysis system through efficient inspection method according to necessity of upgrading system vulnerability," *J. Korea Acad. Ind. Cooperation Soc.* **18**(7), 1–8 (2017).
17. K. H. Kim, D. I. Lee, and Y. T. Shin, "Research on cloud-based on web application malware detection methods," *Lect. Notes Electr. Eng.* **474**, 817–822 (2018).
18. K. Pavani, H. Mishra, and R. Karsh, "Multi-attached network topology with different routing protocols and stub network resolution in OSPF routing," in *Proc. Third Int. Conf. Microelectron., Comput. and Commun. Syst.*, pp. 129–141 (2019).
19. K. Nandhini and R. Balasubramaniam, "Malicious website detection using probabilistic data structure bloom filter," in *3rd Int. Conf. Comput. Methodologies and Commun. (ICCMC)*, Erode, India, pp. 311–316 (2019).
20. S. Kyung-Sang and N. Wonshik, "A study on the implementation of a system providing reliable malware information service," *Int. J. Electr. Eng. Educ.* **58**(2), 517–530 (2019).
21. C. Sharma, S. C. Jain, and A. K. Sharma, "A quantitative risk analysis methodology for the security of web application database against SQL injection (SQLi) attacks utilizing fuzzy logic system as computational technique," *Int. J. Electr. Eng. Educ.* (2019).
22. M. A. Mohammed et al., "Implementing an agent-based multi-natural language anti-spam model," in *Int. Symp. Agent, Multi-Agent Syst. and Rob. (ISAMSR)*, Putrajaya, Malaysia, pp. 1–5 (2018).
23. T. Shibahara et al., "Detecting malicious websites by integrating malicious, benign, and compromised redirection subgraph similarities," in *IEEE 41st Annu. Comput. Software and Appl. Conf. (COMPSAC)*, Turin, Italy, pp. 655–664 (2017).

24. H.-W. Hsiao, D.-N. Chen, and T. J. Wu, "Detecting hiding malicious website using network traffic mining approach," in *2nd Int. Conf. Educ. Technol. and Comput.*, Shanghai, China, pp. V5-276–V5-280 (2010).
25. G. Tan et al., "Adaptive malicious URL detection: learning in the presence of concept drifts," in *17th IEEE Int. Conf. Trust, Security and Privacy in Comput. and Commun./12th IEEE Int. Conf. Big Data Sci. and Eng. (TrustCom/BigDataSE)*, New York, New York, pp. 737–743 (2018).
26. W. Chia-Chun et al., "A trustworthy web-based system platform for teaching evaluation and STEM education," *Int. J. Electr. Eng. Educ.* (2019).

Yong-joon Lee received his PhD in computer science from Soongsil University, in 2005. From 2006 to 2009, he was a deputy researcher at the LG CNS Technology Research Department. From 2010 to 2015, he was a senior research fellow with the Korea Internet and Security. From 2016 to 2019, he was a digital forensic research officer at Information Security Office, Defense Security Support Command, Republic of Korea. He is currently an assistant professor with the Department of Cyber Security, Far East University, Republic of Korea. His research interests include industrial security, cybersecurity, and internal information leakage prevention.

Won-shik Na: Biography is not available.