Lightweight Approach to Detect Drive-by Download Attacks Based on File Type Transition

Yasutaka Shindo
Tokyo Institute of Technology
shindo@netsys.ce.titech.ac.jp

Akihiro Satoh
Kyushu Institute of Technology
satoh@isc.kyutech.ac.jp

Yutaka Nakamura
Kyushu Institute of Technology
yutaka-n@isc.kyutech.ac.jp

Katsuyoshi Iida
Tokyo Institute of Technology
iida@gsic.titech.ac.jp

1. INTRODUCTION

A web-based attack, drive-by download attack, has been posing serious threats to Internet users. Drive-by download attacks have many potential victims because they can target all client browsers on the Internet [1].

Drive-by download attacks often have three malware infection steps [2]. First, when a victim visits a landing site into which malicious code, such as JavaScript or IFRAME, has already been injected by an attacker, and the code makes the victim redirect to some malicious sites. Then, going through some redirections, the victim is eventually led to an attack site that exploits the victim's browser vulnerabilities and compromises the victim's computer. Those vulnerabilities can be PDF, JAR, and Flash in typical cases. Finally, the compromised computer sends requests to malware distribution sites to install malware, such as EXE files, onto the victim's computer.

To detect drive-by download attacks, numerous studies have been published, and their methods have been categorized into two types. One is a code analysis based method. Cova et al. [3] have proposed a method to detect and analyze malicious JavaScript codes by using anomaly detection techniques and dynamic emulation. Another is a rule-based method. Sakai and Sasaki [4] have proposed a rule-based method to detect drive-by download attacks by using particular HTTP header information.

However, code analysis based methods require high analytical costs. Also, rule-based methods have difficulty in forming comprehensive detection rules to keep up with changes in the attack pattern. For network administrators to inspect a huge amount of communication data, a detection method must be developed that requires less costs to describe the attack's distinctive features and less effort to form detection rules.

Our goal is to develop a low-cost and effortless detection method. Focusing on the fact that different specific files are fetched from malicious websites in the three drive-by download steps, we represent the transition of fetched file types by using Content-Type information as lightweight information. The file type transition is converted into a feature vector for a machine learning algorithm that effortlessly creates a classifier to distinguish between malicious and benign communications. In this paper, we propose a lightweight and machine learning-based approach and evaluate its effectiveness.

CoNEXT Student Workshop'14, December 2, 2014, Sydney, Australia.
Copyright 2014 ACM 978-1-4503-3282-8/14/12 ...$15.00.
http://dx.doi.org/10.1145/2680821.2680826.

2. PROPOSED DETECTION METHOD

In the three steps of drive-by download attacks, different specific types of files are fetched through time from malicious websites to a victim. We take advantage of those file types as a drive-by download attack's feature because specific files (such as Flash, PDF, and JAR) are often found in drive-by download attacks. To represent the transition of the file types, we use Content-Type information as lightweight information that can be found in the HTTP response header and describes the payload category contained in the HTTP body.

Figure 1 shows an example process of the proposed method. When a client PC fetches multiple files from malicious websites through time, we consider the time-series order of fetched files as a file type transition. A file type transition belongs to a series of HTTP traffic, a *session*. A Session is all the HTTP traffic when you access an URL. In Figure 1, the sequence of nine file types from the leftmost .html to the rightmost .css is the file type transition in the session. We extract some features from the file type transition, but what kind of features can we extract from it? We take out parts of file type transition based on *suspicious files*, which are files often found in drive-by download attacks, such as JavaScript, Flash, PDF, and JAR. Around a suspicious file, we choose several n files to form a *sample* except for image files that are less likely to contain malicious contents. For instance, four samples have been extracted from the session in Figure 1. Also, we defined n as two in this case, so each sample consists of five file types. Then, we convert each sample into a feature vector to implement machine learning. A feature vector is made up of each file type vector. Each file type vector is described as a dummy variable that has m dimension. m is the number of drive-by download related files that we define in advance.

Following the procedure described above, we collect labeled ('malicious' or 'benign') feature vectors from prerequired

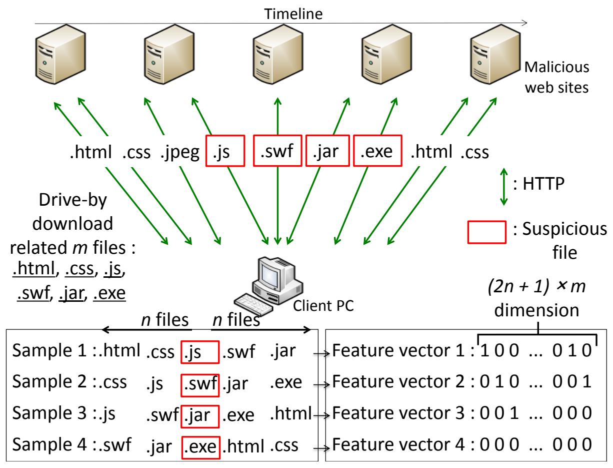
Timeline
Malicious web sites
.html .css .jpeg .js .swf .jar .exe .html .css
: HTTP
: Suspicious file
Drive-by download related *m* files : .html, .css, .js, .swf, .jar, .exe
Client PC
n files
n files
(2n + 1) × m dimension
Sample 1 :.html .css .js .swf .jar
Sample 2 :.css .js .swf .jar .exe
Sample 3 :.js .swf .jar .exe .html
Sample 4 :.swf .jar .exe .html .css
Feature vector 1 : 1 0 0 ... 0 1 0
Feature vector 2 : 0 1 0 ... 0 0 1
Feature vector 3 : 0 0 1 ... 0 0 0
Feature vector 4 : 0 0 0 ... 0 0 0

Figure 1: Example process of proposed method

Table 1: Number of sessions in used datasets.

	Malicious	Benign
Source	D3M	Alexa
All Sessions	200	9911
Sessions (JavaScript)	28	8767
Sessions (Flash)	15	3154

data from some sources and input those vectors into a machine learning algorithm to generate a classifier. Once a classifier has been derived, we can input test data into it and obtain the result showing if the test data is malicious or benign.

3. EVALUATION

We have evaluated the effectiveness of our approach in two types of suspicious files: JavaScript and Flash. In this chapter we explain how we have evaluated our method and briefly discuss the result.

Table 1 shows the number of sessions in used datasets. For malicious data, we used D3M dataset [5]. D3M dataset contains drive-by download attack communication data, which has been collected by NTT Secure Platform Laboratories. For benign data, we crawled Alexa's top 10 thousands sites' URLs. Each piece of data was broken into a session, and we extracted samples from each session.

We defined 12 kinds of file as drive-by download related files and chose two files around a suspicious file when extracting samples. Therefore, feature vectors are $(2 \times 2+1) \times 12 = 60$ dimension. As a machine learning algorithm, we used a Support Vector Machine, one of the most common and efficient methods for binary classification, and performed leave-one-out cross validation.

In both JavaScript and Flash cases, the number of benign sessions and samples are much larger than that of malicious sessions and samples. Thus, we conducted a random sampling of benign sessions and samples so as to have their number be the same as the number of malicious sessions and samples. We conducted this random sampling in 10 patterns and the result was calculated as the average of 10 patterns.

As a result of binary classification, each sample is classified as 'malicious' or 'benign'. In some cases, multiple samples come from a session, and some of the samples are classified as malicious while the others are classified as benign. How can you tell if the session is malicious or benign? In this experiment, we can tell a session is malicious if at least one sample is classified as malicious.

Table 2: Detection rate

	JavaScript	Flash
Detection rate average [%]	81.8	48.1
Standard deviation	1.8	21.3

Table 3: Incorrectly classified samples

	Session ID	Malicious sample
JavaScript	1	.js .js .js .js .js
		.js .js .js .pdf .js
		.js .pdf .js .js .css
Flash	2	.html .html .swf .swf .swf
	3	.swf .js .swf .html .swf
	4	.html .pdf .swf .jar .jar

The results are shown in Table 2. We achieved a decent result, 81.8 %, in the case of JavaScript files. On the other hand, the result with Flash files was 48.1 % at most.

Table 3 shows some misclassified malicious samples. They are incorrectly labeled as 'benign', which means that those malicious samples have something in common with benign samples. There are thought to be two reasons for some of the misclassified malicious samples. One is that there are samples that consist of some of the same file types (Sessions 1, 2, 3). For benign data, we used Alexa's top sites, which include a lot of popular commercial websites. Most of them include a bunch of the same kind of files, so each benign session is likely to include a lot of the same kind of files. Therefore, we obtained those samples as benign samples, which led to this type of malicious samples being misjudged. The other one is that a small number of samples is ineffective to derive a decent classifier (Session 3). Even though the sample includes PDF, Flash, and JAR file types that are characteristic of malicious samples, a sufficient number of samples is needed to derive a sophisticated classifier to correctly classify such a distinctive sample.

4. CONCLUSIONS

We presented a lightweight and machine learning-based approach to distinguish between malicious and benign communications. Focusing on the fact that different specific files are fetched from malicious websites in drive-by download steps, we represent the transition of fetched file types by using Content-Type information in a series of HTTP traffic. We conducted experiments with JavaScript and Flash files as suspicious files. Consequently, we achieved a decent result, 81.8 %, in the case of JavaScript files. On the other hand, the result with Flash files was 48.1 % at most. Some malicious samples were misclassified because they consist of the same file types. Also, there were not enough malicious samples to derive an effective classifier.

Our future work is to look into the detailed reasons the experiment resulted in the low score with Flash files. After that, we will need to come up with some ways to improve the detection rate. Also, we will need to conduct experiments with the other drive-by download related files, such as PDF and JAR.

5. REFERENCES

[1] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose, "All Your iFRAMEs Point to Us," Proc. USENIX Conference on Security Symposium, Feb. 2008.

[2] J. Chang, K. K. Venkatasubramanian, and A. G. West, "Analyzing and defending against web-based malware," ACM Computing Surveys (CSUR), vol. 45, no. 4, pp. 49:1–49:35, Aug. 2013.

[3] M. Cova, C. Kruegel, and G. Vigna, "Detection and analysis of drive-by-download attacks and malicious JavaScript code," Proc. International Conference on World Wide Web, pp. 281–290, Apr. 2010.

[4] H. Sakai, and R. Sasaki, "Proposal of detection method based on HTTP headers against Drive By Download Attack (in Japanese)," IPSJ SIG-DPS Technical Report, vol. 2013, no. 29, pp. 1–6, Mar. 2013.

[5] M. Kamizono, M. Hatada, M. Terada, M. Akiyama, T. Kasama, and J. Murakami, "Datasets for Anti-Malware Research — MWS Datasets 2013 — (in Japanese)," Proc. IPSJ Computer Security Symposium 2013 (CSS2013), vol. 2013, no. 4, pp. 1–8, Oct. 2013.