2015 10th Asia Joint Conference on Information Security

An Approach to Predict Drive-by-Download Attacks by Vulnerability Evaluation and Opcode

Takashi Adachi
Japan Advanced Institute of
Science and Technology,
1-1 Asahidai, Nomi, Ishikawa Japan

Kazumasa Omote
Japan Advanced Institute of
Science and Technology,
1-1 Asahidai, Nomi, Ishikawa Japan

***Abstract*—Drive-by-download attacks exploit vulnerabilities in web browsers, and users are unnoticeably downloading malware which accesses to the compromised websites. A number of detection approaches and tools against such attacks have been proposed so far. Especially, it is becoming easy to specify vulnerabilities of attacks, because researchers well analyze the trend of various attacks. Unfortunately, in the previous schemes, vulnerability information has not been used in the detection/prediction approaches of drive-by-download attacks.**
In this paper, we propose a prediction approach of "malware downloading" during drive-by-download attacks (approach-I), which uses vulnerability information. Our experimental results show our approach-I achieves the prediction rate (accuracy) of 92%, FNR of 15% and FPR of 1.0% using Naive Bayes. Furthermore, we propose an enhanced approach (approach-II) which embeds Opcode analysis (dynamic analysis) into our approach-I (static approach). We implement our approach-I and II, and compare the three approaches (approach-I, II and Opcode approaches) using the same datasets in our experiment. As a result, our approach-II has the prediction rate of 92%, and improves FNR to 11% using Random Forest, compared with our approach-I.

Keywords—Drive-by-Download Attacks, Malware, Supervised Machine Learning

I. Introduction

In recent years, drive-by-download attacks exploit vulnerabilities in web browsers and a number of attacks have been increased. We show the flow of such attacks in Figure 1. In the first step, adversaries compromise legitimate websites, and then inject malicious codes into them to redirect client devices to the relay servers. The relay server detects the fingerprinting of client devices (e.g., OS, the version of plugin and user-agent). If it is possible to exploit vulnerabilities of client devices by the fingerprinting, the client devices are redirected to malware distribution server. If it fails to detect vulnerabilities, then client devices are redirected/displayed to the benign websites. If the attack succeeds, then malware is downloaded to the client devices.

These attacks exploit vulnerabilities of plugins or OS in order to make client devices download malware from the malicious web sites. Of course, the best preventive measure is security-patches of client devices, but many users do not apply security-patches owing to the lack of risk management. Actually, users of 80% do not apply patches [2]. Hence we need an approach to detect malicious web pages (e.g., Google Safe Browsing). Many researchers focus on JavaScript or HTML to detect drive-by download attacks. Unfortunately, the

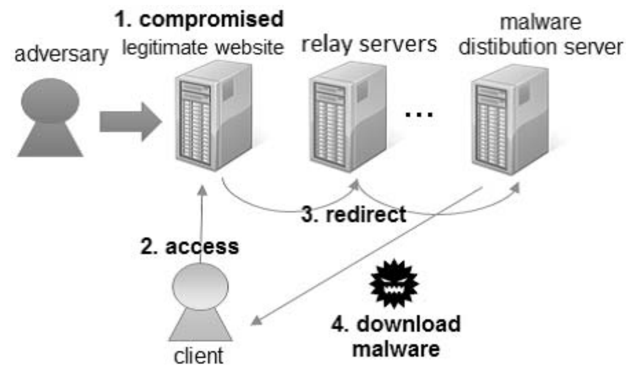


Fig. 1: An overview of drive-by-download attacks

programing-language like JavaScript is usually obfuscated by adversaries. The obfuscated or evasive codes is emerging in real attacks, and thus it is important to use robustness features. So we employ exploiting vulnerabilities during an attack as a robustness feature. The exploiting vulnerabilities have not been used to detect malicious web pages so far, although this information is very important when we detect such attacks.

In this paper, we propose an prediction approach of "malware downloading" during drive-by-download attacks (approach-I), which uses vulnerability information. This approach has an advantage: being tolerant for variety and progress of attacks. Our experimental results show the prediction rate (accuracy) of more than 85% of our approach-I. This approach employs 500 of benign data and D3M datasets which contain malicious web pages and scripts to evaluate our method. However, our approach-I has two disadvantages that False Negative Rate (FNR) is not low, and detecting attacks is not possible when our approach-I does not find vulnerabilities.

We also propose an approach-II in order to mitigate the above disadvantages of approach-I. The enhanced approach (approach-II) embeds Opcode analysis (dynamic approach) into our approach-I (static analysis). Dynamic analysis uses in-between language (Opcode) with JavaScript. This approach also employs 500 of benign data and D3M datasets. We implement these approaches and demonstrate the efficacy of our approaches in the experiments. As a result, our approach-II has the prediction rate of 92%, and improves FNR to 11% using Random Forest to mitigate the disadvantages of approach-I.

This paper is structured as follows. Section II presents prior works to detect the attacks. Section III describes some useful analysis tools/services and dataset as a preliminary. Section IV introduces our approach-I in detail and Section V describes the enhanced approach (approach-II) that embeds Opcode analysis (dynamic approach) into our approach-I (static approach). Section VI evaluates our approaches and Opcode analysis, Section VII discusses our approaches, and Section VIII concludes this paper.

II. Related Work

A. Vulnerabilities

Frigault et al. [11] evaluates the vulnerabilities of network by scan and then prevents infections of adversaries in advance. Xie et al. [12] predict infection routes of adversaries by Bayesian Networks. The vulnerabilities have been used in the evaluation of network.

There are two works against the drive-by-download attacks. One analyzes the trend of such attacks and the other analyzes the exploited vulnerabilities in malicious pages. These works are opened as the analysis service/tool (wepawet [14] and jsunpack [5]) of malware. The wepawet is the analysis service which includes JSAND as the detection system [10]. JSAND analyzes JavaScript using HtmlUnit, monitors DOM tree, function call and behavior to detect attacks, and specifies the vulnerabilities. The jsunpack is the analysis tool that can deobfuscate JavaScript, specify vulnerabilities and reconstruct the HTTP sequence [5]. In particular, it is easy to specify the vulnerabilities of attacks in recently. Unfortunately, we have not been found the prediction/detection approaches using vulnerability information.

B. Prediction and Detection

The prediction/detection approaches for drive-by-download attacks are classified three methods which consist of the offline, the semi realtime and the realtime methods. The offline methods often employ dynamic analysis. EVILSEED [4] performs cloaking to websites and analyze them in a server, and then it adds web pages to the blacklist if they are malicious. The method [10] performs dynamic analysis to malicious web pages using the browser emulator and then it detects the attacks using features such as the number of redirections and executions. These offline methods have an advantage that there is a little restriction of a resource, but they have the disadvantages such as cloaking, finger-printing and costs.

Semi realtime method is embedded in a server infrastructure such as a proxy server. This approach detects attacks during the session of browser. The representative approach is CUJO developed by Rieck et al. [9] whose method employs the combination of static and dynamic analyses. Static analysis replaces variables or characters with simple words by JavaScript Lexer and then extract features from them. Dynamic analysis uses an operation of JavaScript, and then extracts features. This approach determines malicious if either static analysis or dynamic analysis detects malicious. Semi-realtime approach improves the disadvantages of offline methods, but it is affected by the influences of user behavior or executing path.

Realtime approach includes Browser JS Guard developed by Kishore et al. [8] and Opcode analysis by Jayasinghe [7]. Browser JS Guard deobfuscates JavaScript from the extension of browser by HTML tags and hooking process. Additionally, this approach detects the hided iframe or other suspected tags, and then it generates the user-alert if web pages are malicious. Opcode analysis approach logs Opcode with JavaScript which is created by SpiderMonkey. It extracts features using N-gram and then builds a detection model. This approach has a high detection rate and efficient, but this approach depends on JavaScript. The realtime approach has the strict of resources and the heavy computational calculation. However, this approach has the advantages: we do not have to consider cloaking or fingerprinting and can obtain malicious codes from the browser.

III. Preliminary

A. HtmlUnit [3]

HtmlUnit is the browser emulator which is based on Java for testing web-applications. It supports JavaScript by Mozilla's Rhino interpreter. This tool can emulate a modern browser such as Firefox or Internet Explorer and can easily configure the HTTP headers, Active X control and plugins. It also includes anti-cloaking technology. We employ this tool in the dynamic analysis.

B. SpiderMonkey [13]

SpiderMonkey is the first JavaScript engine that was developed and managed by Mozilla Foundation. Firefox browser includes SpiderMonkey which parses and executes JavaScript. We can built SpiderMonekey in arbitrary applications and customize it due to Open Source Software (OSS).

C. Wepawet [14]

Wepawet is the online analysis tool of malware that was developed by California University. This tool consists of JSAND which analyzes JavaScript in malicious sites using HtmlUnit. We can obtain the exploited vulnerabilities and the deobfuscation JavaScript in malicious websites using this tool. Even if the malicious URL is changed into benign websites now, this tool returns the exploited vulnerabilities that this tool investigated in the past.

D. D3M Datasets [1]

D3M datasets consist of the malicious web pages and the scripts gathered on the web-based client honeypot with high-interaction, which is managed by NTT Secure Platform Laboratories. The dataset include malware, the network monitoring data for malware execution on Sandbox, and full capture traffics when such honeypot accesses to the malicious URL.

IV. Our Approach (Approach-I)

Most studies against drive-by download attacks focus on the programing-language like JavaScript or HTML as a feature. However, the programing-language like JavaScript is usually obfuscated by adversaries. Rieck et al. [10] and Clements et al. [6] indicate that the browser emulator does not read images in web pages but a real browser reads them, and thus the

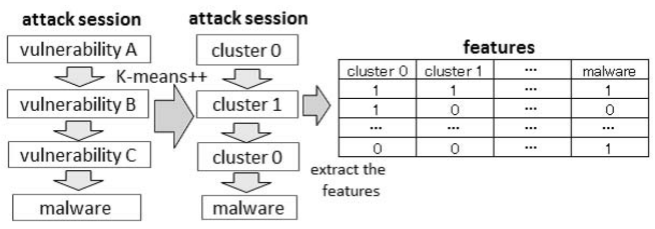


Fig. 2: Feature extraction in our approach-I

behavior of these browsers is different in processing error. So we need an essential feature tolerant against such behavior. Therefore, we propose an prediction approach of malware downloading during drive-by-download attacks (approach-I), which uses vulnerability information. Hence this approach has an advantage that it does not have the strict of programing-languages.

Our approach-I predicts malware downloading by evaluating vulnerabilities in malicious pages. To evaluate the vulnerabilities, we need to identify CVE-IDs in the page. The wepawet can identify CVE-IDs from the web page, hence we employ the wepawet as the analysis tool and then we obtain information of CVEs from National Vulnerability Database (NVD). The part of detected CVEs has similar properties. In generally, unnecessary information makes detection rates decrease. Therefore, we reduce unnecessary information due to apply grouping algorithm. We extract features from excepted information, then a prediction model is built using features extraction. The prediction model calculates a probability of malware downloading after we detect malicious pages on the website. We have three phases in the following paragraphs.

A. *Preparation Phase*

This phase consists of two steps. In the first step, we obtain CVE-IDs from malicious pages using wepawet. We obtain the detailed information of each CVE-ID from NVD Database. The detailed information consists of basic seven factors which include “Factor”, “Score”, “AccessVector”, “Authentication”, “Confidentiality”, “Integrity”, and “Availability”. We can receive other information but it does not always exist. We perform clustering to each CVE-ID using K-means++ algorithm.

In the second step, we extract features described in Figure 2. The left subfigure shows that this method extracts vulnerabilities used in the attack session. We apply clustering algorithm to left subfigure, and then the center subfigure is generated. This center subfigure shows the extraction features except unnecessary information. In the right subfigure, we extract features from the center subfigure. For example, 1 of cluster 0 means that we detect a group of vulnerability once, 0 of cluster 1 means that we do not detect vulnerabilities. If a group of vulnerability is exploited more than twice, then the features do not include twice. We extract the features by detecting a group of vulnerability, and then occurring of malware downloading is added. The rows in the features table are the number of attacks and the cows are the number of clusters and the appearance of malware downloading.

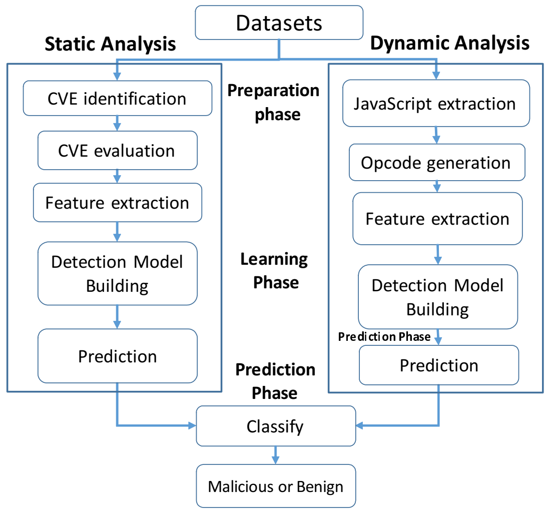


Fig. 3: A schematic view of enhanced approach (approach-II)

B. *Training Phase*

The feature vectors are learned by the supervised machine learning algorithms to obtain the prediction model for drive-by-download attacks.

C. *Prediction Phase*

We need to extract the features through each attack session. In the first step, we detect vulnerabilities of each attack session using wepawet. In the second step, we obtain the detailed information of each CVE and apply clustering algorithm to them. In the final step, we extract the features from results of clustering algorithm and then calculate the prediction probability of malware downloading. If the prediction probability is more than 0.5, the prediction model determines that malware is downloaded by a drive-by-download.

V. Enhanced Approach (Approach-II)

Our approach-I is tolerant for progress of attacks, because the vulnerability is necessarily used in drive-by-download attacks. Additionally, the approach-I has an advantage that it does not have the strict of program-language. However, our approach-I has a disadvantage: it cannot predict malware downloading if we do not detect vulnerabilities in malicious pages. Thereby, we enhance our approach-I using in-between language (Opcode) with JavaScript. Jayasinghe et al. [7] indicates that Opcode is an efficient and effective feature, and this approach has the high detection rates.

The Opcode approach has a disadvantage that we cannot detect attacks which do not employ JavaScript. In order to mitigate the above both disadvantages, we propose an enhanced approach (approach-II) that combines our approach-I and Opcode approach. This approach can utilize the advantages of both approaches.

A schematic view of enhanced approach is presented in Figure 3. This approach consists of our approach-I (static

TABLE I: Example of Opcode Features

setlocal	pop	getlocal	dup	...	malware
1	3	5	2	...	1
3	2	1	4	...	0
⋮	⋮	⋮	⋮	...	⋮
0	2	2	7	0	1

approach) and Opcode approach (dynamic approach). In the preparation phase, the features are extracted from malicious pages. In the training phase, we build a prediction model for drive-by-download attacks using the extracted features and the supervised machine learning algorithm. In the prediction phase, we extract features from test data, and then we evaluate the prediction probability of malware downloading. We need to choose the threshold of prediction probability. This border is determined by CUJO and preliminary experiments, more precisely, the threshold is higher than 0.5 of each approach.

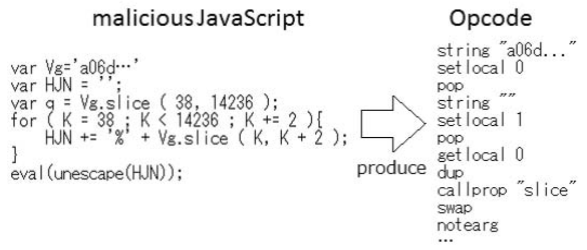
Dynamic analysis detects attacks using Opcode with JavaScript. This approach needs to extract JavaScript from the datasets. We can produce Opcode with JavaScript by SpiderMonkey which is interpreter. Additionally, we calculate frequency of Opcode into one attack session by the extracted instruction codes. The frequency of Opcode becomes the features. The prediction model is built by the supervised machine learning. This approach calculates the prediction probability of malware downloading when we detect JavaScript.

A. Preparation Phase

The static analysis part is the same as our approach-I. On the other hand, the preparation phase of dynamic analysis has three steps. In the first step, we extract JavaScript from the web pages in the script tags or a file referred by src-property in the dataset. Malicious codes include benign JavaScript or non-JavaScript code, but we do not process them. In the second step, SpiderMonkey produces Opcode from JavaScript described in Figure 4. The extracted JavaScript includes DOM (e.g., document.write) which is the browser operations, but we do not process them too. Opcode has the instruction codes, the first operand, the second operand and the other information. CUJO employs operands as features, but Opcode analysis employ instruction coded only [7]. We extract instruction codes only because an analysis time increases. In the third step, we calculate a frequency of each word from instruction codes extracted. Additionally, we create the feature vectors as illustrated in Table I. To create feature vectors, frequency of instruction codes of each session and information of malware downloading is aggregated, and then we create the matrix of the features vector. The rows of features are number of attack sessions and the cows are the number of kinds of Opcode. In our experiments, the number of kinds of Opcode is 130 that except "unused".

B. Training Phase

We build a prediction model for drive-by-download attacks using the extracted features and the supervised machine learning algorithms by static and dynamic analysis.



```
var Vg='a06d…'
var HJN = '';
var q = Vg.slice ( 38, 14236 );
for ( K = 38 ; K < 14236 ; K += 2 ){
   HJN += '%' + Vg.slice ( K, K + 2 );
}
eval(unescape(HJN));
```

```
string "a06d..."
setlocal 0
pop
string ""
setlocal 1
pop
getlocal 0
dup
callprop "slice"
swap
notearg
...
```

Fig. 4: Produce Opcode from JavaScript

C. Prediction Phase

Static analysis in the prediction phase is the same as our approach-I. On the other hand, dynamic analysis has three steps. In the first step we extract JavaScript from each attack session of the datasets. In the second step, we produce Opcode from JavaScript using SpiderMonkey. In the third step, we calculate the frequency of Opcode and then it extracts the features. We calculate the prediction probability of malware downloading using the features in both our approach-I and Opcode approach. The calculated probability is between 0 and 1, and 0 means that malware is not downloaded. If one of the prediction probabilities of our approach-I and Opcode approach is higher than 0.5, the prediction model determines that malware is downloaded by a drive-by-download.

VI. Experimental Evaluation

Our approach uses benign and malicious web pages. The benign pages are collected from AlexaTop500 using HtmlUnit. The malicious pages employ the D3M datasets of 2011-2014. We divide all pages into training and test data as follows.

- Training data:
  the half of the random benign pages and the malicious pages of 2011 year.
- Test data:
  the half of the rest benign pages and the malicious pages of 2012–2014 years.

The training data applies for building the prediction model. We apply the following supervised machine learning algorithm: Naive Bayes, SVM, LinearSVC, Decision Tree and Random Forest. SVM and LinearSVC is similar algorithm, however kernel of algorithm is a difference.

The testing data uses for evaluating the approaches. As illustrated in Table II, we employ the five kinds of supervised machine learning algorithms with kernel. The approaches are implemented and evaluated using scikit-learn of Python. To evaluate our approach-I, we repeat experiments 10 times and use the average results. The results include True Positive (TP), False Positive (FP), False Negative (FN), True Negative (TN) and Accuracy (detection rate). We implement approaches that include our approach-I, II and Opcode approach.

A. Approach-I

1) Evaluation: We implement and experiment our approach-I to show the effective of vulnerabilities used in the

TABLE II: Algorithm and kernel of the supervised machine learning

Algorithm	Static Analysis	Dynamic Analysis	Remarks
Naive Bayes	bernoulli distribution	same static	prior probability is calculated by datasets
SVM	poly	poly	degree of poly kernel is 1
LinearSVC	loss and penalty is L2	same static	N/A
Decision Tree	entropy	entropy	N/A
Random Forest	entropy	entropy	N/A

TABLE III: Prediction rates of approach-I

	TPR	FPR	FNR	TNR	ACC
Naive Bayes	0.85	0.01	0.15	0.99	0.92
SVM	0.79	0.01	0.21	0.98	0.89
LinearSVC	0.68	0.06	0.32	0.93	0.85
Decision Tree	0.67	0.00	0.33	1.00	0.83
Random Forest	0.77	0.01	0.23	0.99	0.88

TABLE IV: Prediction rates of Opcode approach

	TPR	FPR	FNR	TNR	ACC
Naive Bayes	0.87	0.25	0.13	0.75	0.81
SVM	0.82	0.14	0.18	0.86	0.84
LinearSVC	0.68	0.06	0.32	0.93	0.80
Decision Tree	0.82	0.10	0.18	0.90	0.86
Random Forest	0.85	0.06	0.15	0.94	0.90

drive-by-download attacks. In the experiment, it performs for tenth, then the average of results become prediction rates. Note that the number of clusters is set to 10.

2) Results: We show the experimental results of our approach-I in Table III, in which the prediction rates of approach-I is higher than 83%. Therefore, the vulnerability information is an effective feature against attacks. The identified vulnerabilities in the datasets are 28.

3) False Positive Rates and False Negative Rates: In this approach, FPR is low because analysis tool does not detect vulnerabilities in benign pages, and also the prediction probability of malware downloading is low. However, the average of FNR is not low. This is caused by the following two reasons; (1) we cannot detect the vulnerabilities in malicious pages and (2) the detected vulnerabilities are not much threat compare with other vulnerabilities. In the first reason, we do not evaluate vulnerabilities and then the prediction model does not calculate the prediction probability of malware downloading. In the second reason, adversaries tend to employ the specific vulnerabilities which has the high threat and hence it may be difficult to predict malware downloading using vulnerabilities with low threat. Thereby, we cannot predict the malware downloading.

B. Approach-II

1) Evaluation: We implement our approach-II, and then compare it with two approaches which are our approach-I and Opcode approach.

2) Results: We show the experimental results of our approach-I, Opcode approach and our approach-II in Table III, IV and V, respectively. As a result, our approach-II is the best model that combines static and dynamic analysis. Especially, our approach-II improves FNR of about 30% compared with our approach-I using Naive Bayes or Opcode approach using Random Forest.

TABLE V: Prediction rates of approach-II

	TPR	FPR	FNR	TNR	ACC
Naive Bayes	0.89	0.25	0.11	0.75	0.82
SVM	0.87	0.13	0.13	0.87	0.87
LinearSVC	0.73	0.06	0.27	0.93	0.83
Decision Tree	0.84	0.10	0.16	0.90	0.87
Random Forest	0.89	0.06	0.11	0.94	0.92

3) False Positive Rates and False Negative Rates: Table VI shows the prediction probabilities of benign sessions which are determined incorrectly as a malicious session. Our approach-I calculates a low prediction probability of malware downloading since the vulnerability is detected in the benign session 1 and 2. On the other hand, the Opcode approach has the high prediction probability of malware downloading since these benign sessions include many Opcode which is often used in malicious pages. Therefore, the Opcode approach determined these benign session incorrectly as a malicious.

We show that our approach-II fails to predict malware downloading in attack sessions of Table VII. An attack session means a series of drive-by-download attacks. In the session 1 and 2, both our approach-I and Opcode approach cannot detect malicious pages, because our approach-I cannot detect vulnerabilities. Furthermore, Opcode approach fails to predict malware downloading because these sessions consist of many benign JavaScript and few malicious pages. The session 3 and 4 also fail to predict malware downloading even if our approach-I detects vulnerabilities in malicious pages. The prediction probability of malware downloading becomes low since the detected vulnerabilities do not have much threat. Actually, most attacks employ the high threat vulnerabilities. The Opcode approach cannot detect malicious pages because the few malicious scripts are hidden in benign scripts.

TABLE VI: Prediction probabilities of benign sessions which are determined incorrectly as a malicious session

benign session	prob. of approach-I	prob. of Opcode
1	0.31	0.84
2	0.31	1.00

TABLE VII: Prediction probabilities of attack sessions which are determined incorrectly as a benign session

attack session	prob. of approach-I	prob. of Opcode
1	0.00	0.31
2	0.07	0.31
3	0.49	0.31
4	0.48	0.31

VII. Discussion

A. Comparison of Approaches

The FPR of our approach-I is low compared with the other approaches, and thus this suitable for identification of benign pages. On the other hand, the FNR of Opcode analysis is low in average compared with our approach-I. Therefore, the Opcode approach is suitable for identification of malicious websites. However, the FPR of Opcode approach is high compared with our approach-I because there are malicious pages which include many benign scripts and a few malicious scripts. Our approach-II achieves low FNR compared with the other approaches due to the supplement of each advantage.

B. Vulnerabilities

In generally, the exploiting vulnerabilities have been changed with time because many researchers find a new vulnerability. So it is important to perform the grouping of the vulnerability with a feature. Even if vulnerability changes, the feature of vulnerability used in the attack would not change. The effect of a grouping is expectable.

For the effective grouping, we need to evaluate quantitatively the vulnerabilities. To evaluate them quantitatively, this approach employs the clustering algorithm. Therefore, we verify whether the clustering algorithm is effective or not by the experiment. We show the effectiveness of grouping in Table VIII which is the results by Naive Bayes. This table indicates that the grouping by clustering algorithm is effective. Surprisingly, the results shows that the accuracy is improved 50% when we apply the clustering algorithm to the preparation phase in our apporach-I. Additionally, 28 CVEs are classified into 10 groups by clustering. As a result, the similar CVEs become the same group and unnecessary information can be deleted. The clustering algorithm makes general information of CVEs (whether it or not) adapt changing of vulnerabilities. Consequently, our approach-I becomes to have the high prediction rate due to clustering.

TABLE VIII: Effectiveness of grouping (Naive Bayes)

	None	Grouping
TPR	0.83	0.85
FPR	0.89	0.01
FNR	0.17	0.15
TNR	0.11	0.99
Accuracy	0.47	0.92

TABLE IX: Top 5 biggest difference list of Opcode ratio between benign and malicious pages in the datasets

Opcode	benign	malicious
initprop	0.87	0.21
true	0.85	0.20
false	0.85	0.21
newobject	0.88	0.25
typeofexpr	0.74	0.13

C. Opcode

Table IX shows the Top 5 biggest difference list of Opcode ratio between benign and malicious pages in the datasets. For example, the "initprop" of 87% is used in benign pages, meanwhile that of 21% is used in malicious pages. We consider that the bigger difference is an excellent feature. The "initprop" is used in associative array and thus indicates that it is not used frequently in malicious pages. The malicious pages have a little chance to use an associative array because there is no meaning using the associative array for the obfuscated JavaScripts. According to Boolean value, malicious pages do not need to consider readability or maintainability due to employ the Exploit Kit. On the other hand, the benign pages need readability or maintainability to decrease the human error. Thereby, the benign pages use Boolean values for error detection or input checking, but the malicious pages do not use them. The "typeofexpr" is not used in malicious pages because this also uses Boolean values.

D. Approach-II

Our approach-II has the prediction rate of 92% and FNR of 11% using Random Forest. The approach compares with our approach-I, accuracy is improved by Opcode approach. Especially, this approach decreases FNR to about 50%. In particularly, our approach-II improves FNR and accuracy due to complement each advantage.

VIII. Conclusions

This paper proposes two approaches which predict malware downloading during drive-by-download attacks. This approach focuses on vulnerability information used in such attacks. Our apporach-I achieves the prediction rate of 92%, FNR of 15% and FPR of 1.0% using Naive Bayes. The advantages of this approach are tolerant for variety and progress of attacks. Furthermore, we propose an enhanced approach (approach-II) which embeds Opcode analysis into our approach-I. However, our approach-I has a disadvantage that FNR is not low. On the other hand, Opcode approach has a disadvantage when malicious pages do not include JavaScript. Our apporach-II achieves the prediction rate of 92%, FNR of 11% and FPR of 6.0% using Random Forest. In the future, we plan to improve time to identify vulnerabilities since actually our approaches employs the wepawet tool to identify the exploited vulnerabilities used in drive-by download attacks.

References

[1] A. Mitsuaki, K. Masaki, M. Takashiro, H. Mitsuhiro, “Datasets for Anti-Malware Research -MWS Datasets 2014-”, Computer Security vol.2014-CSEC-66, No. 19.

[2] D. Danchev, “Research:80% of Web users running unpatched versions of Flash/Acrobat”, ZDNet, http://www.zdnet.com/blog/security/research-80-of-web-users-running-unpatched-versions-of-flashacrobat/4097.

[3] HtmlUnit, http://htmlunit.sourceforge.net/.

[4] I. Luca, M. Paolu, “EVILSEED: A Guided Approch to Finding Malicious Web Pages”, IEEE symposium on Security and Privacy, 2012.

[5] jsunpack-n, http://jsunpack.jeek.org/.

[6] K. Clements, L. Benjamin, Z. Benjamin, S. Christian, “ROZZLE: De-Cloaking Internet Malware”, Microsoft Reseach, 2011.

[7] K. Jayasinghe, J. Shane, P. Bertok, “Efficient and effective realtime prediction of drive-by-download attacks”, Journal of Network and Computer Applications 38, 2014.

[8] K. Kishore, M. Mallesh, G. Jyostna, P R L Eswari, S. Samavedam, “Browser JS Guard: Detects and Defends against Malicious JavaScript Injection based Drive by Download Attacks”, Applications of Digital Information and Web Technologies (ICADIWT), 2014.

[9] K. Rieck, T. Krueger, A. Dewald, “Cujo: Efficient Detection and Prevention of Drive-by-Download Attacks”, In Proceeding of the 26th annual computer security applications conference (ACSAC ’10), 2010.

[10] M. Cova, C. Kruegel, G. Vigna, “Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code”, In Proceeding of the 19th international conference on World Wide Web (WWW ’10), 2010.

[11] M. Frigault and L. Wang, “Measuring Network Security Using Bayesian Network-Based Attack Graphs”, 32nd Annual IEEE International Computer Software and Applications Conference, 2008.

[12] P. Xie, J. H Li, X. Ou, P. Liu and R. Levy, “Using Bayesian Networks for Cyber Security Analysis”, 2010 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2010.

[13] SpiderMonkey, https://developer.mozilla.org/en-US/docs/Mozilla/Projects/SpiderMonkey.

[14] wepawet, https://wepawet.iseclab.org/.