情報理論的指標と異常検知に基づく 難読化悪性 JavaScript 検知手法の提案

蘇 佳偉† 吉岡 克成‡ 四方 順司‡ 松本 勉‡

†横浜国立大学大学院環境情報学府
240-8501 神奈川県横浜市保土ケ谷区常盤台 79 番 7 号

‡横浜国立大学大学院環境情報研究院/先端科学高等研究院
240-8501 神奈川県横浜市保土ケ谷区常盤台 79 番 7 号

あらまし Drive-by-download攻撃を起こすための悪性JavaScriptは、セキュリティシステムの検知を回避するために難読化される場合が多い。本研究では、従来の難読化悪性JavaScript検知手法の短所を改善し、情報理論と異常検知に基づく新しい難読化悪性JavaScript検知方法を提案する。

Detecting obfuscated malicious JavaScript based on information-theoretic measures and novelty detection

Jiawei Su† Katsunari Yoshioka‡ Junji Shikata‡ Tsutomu Matsumoto‡

†Graduate School of Environment and Information Sciences, Yokohama National University
79-7 Tokiwadai, Hodogaya-ku, Yokomaha-shi, Kanagawa 240-8501, JAPAN

‡Graduate School of Environment and Information Sciences / Institute of Advanced Sciences, Yokohama National University
79-7 Tokiwadai, Hodogaya-ku, Yokomaha-shi, Kanagawa 240-8501, JAPAN

Abstract The malicious JavaScript is a main medium for computer network attackers to launch popular Drive-by-download attacks. In a typical case, attackers compromise legitimate websites and inject malicious JavaScript into their webpages which is used to bounce the visitors to other preset malicious pages where the visitors actually get infected. In order to evade automatic detectors, attackers always process their malicious JavaScript with varied obfuscation programs so that the actual contents of the original script can be hid. In this paper, we propose a new light-weight filter system for detection obfuscated malicious JavaScript, which improves several critical potential weaknesses of previous analogous detection systems.

I. INTRODUCTION

The investigation domain of detecting malicious JavaScript is always focused by security researchers. Traditional signature matching has been proved as inefficient to against recent malicious JavaScript. Attackers commonly obfuscate their malicious codes by varied or customized algorithms to hide malicious contents so that detectors based on signature matching could be evaded. This pushed researchers to probe new approaches to against such threats. Recently, some researchers have begun to attempt to build new static detectors by utilizing machine learning and statistical technique. In general, the machine learning based detectors could make the detection become fuzzy and flexible, which are extremely good at discovering variants of existing maliciousness as well as unknown threats. Such systems are always proposed as front-end filter for large scale analysis which the filter can rapidly discard potential benign JavaScript and bounce the suspects to a back-end sophisticated analysis system whose detection is more accurate but seriously resource and time intensive. By utilizing such filters, only suspicious JavaScript will be analyzed in detail therefore the resources required for large scale analysis can be substantially diminished.

In this paper, we introduce a new malicious JavaScript filter. Similar to previous analogous systems, it is a light-weight static system but equipped with new extracted features based on information theory, and new system modelling approach based on novelty detection (semi-supervised learning) technique.

It is a common viewpoint that attackers use obfuscation on their malicious JavaScript to hide exploits and prevent exact rules or signature based systems from detecting the attack. According to this circumstance, most of the previous works regarded obfuscation as potential maliciousness and extracted features to characterize it. Namely, most of the "malicious" features that have been extracted are actually used to capture the obfuscation. According to our recent observation, this assumption was again validated.

It is true that some benign JavaScript is also obfuscated whereas malicious JavaScript may also come with plaintext. However, a system we aim to design will be operated over large-scale to detect suspicious JavaScript within a large amount of web page documents, therefore we only focus on the general viewpoint, namely the maliciousness always comes with obfuscation, but not partial exceptions. In addition, one could tune the sensitivity of our system by feeding different parameters. For example, according to the scenario of deploying our system as a front-end filter, the false-positive (false alarm) is less expensive as it only results in waste of resources whereas false-negative (miss detection) will incur exposure to malicious JavaScript and hence it is more critical. Under such situation, one may wish to tune the system to minimize the false-negative so that the system will aggressively judge a given JavaScript as obfuscation. Our system satisfies this requirement well, which allows to be tune into the mode of favoring false-negative, while giving low false-positive rate.

II. RELATED WORK

The machine learning and data mining technique is becoming more and more common for detecting network maliciousness. For instance, in the domain of network intrusion detection, machine learning classifiers such as SVM, neural network and cluster technique have been used as common tools to conduct outlier detection. On the other hand, in the field of detecting malicious programming codes, as most of the programming codes can be processed as pure text streams, or natural languages, it is feasible to utilize document processing techniques to characterize the malicious codes and conduct the classification. Under such background, the following related investigations have been proposed.

A. *Support Vector Machine (SVM) based detection systems*

Davide et al.[1] proposed a fast filter based on SVM to detect maliciousness including malicious JavaScript in web page documents over large scale, which their motivation and background of investigation is very similar to ours. They pointed out the weaknesses of traditional rule-based detection systems as well as the need of a fast maliciousness filter. By using SVM, their new static system achieved elastic and fuzzy detection so that their detection rate approximated to the sophisticated systems while kept the advantage of time and resource efficient. Other analogous SVM-based systems such as [2, 3, 10] which used for directly scanning malicious JavaScript have also been introduced. Within these investigations, researchers extracted a ton of rough and direct 'malicious' features without evaluating their effectiveness and caring the seriously increased amount of features, and attempt to cover all possible malicious behaviors with these trivial features, hence most of these features are not to be guaranteed to have low redundancy and high robustness. For instance, in [1, 2], researchers admitted that their features may have potential robustness problem. On the other hand, the great numbers of trivial features also incurred increasing of dimensions of feature vectors for describing data points (e.g. in [2], there are 65 features and in [10], there are more than 150, compare to our 7 features) which definitely downgrade the running speed of machine learning classifier as well as waste of system resources. In addition, inefficient features may also bring additional noises and confuse the classifiers to carry out a worse detection result.

According to the discussion above, it is necessary to do concentration and refining in order to reduce the dimensions of features without sacrificing the accuracy of detection. This is also a common task called "feature extraction" in machine learning, which requires the extracted features are concentrated and as less as possible.

B. Approach based on frequency

Although [4, 5] proposed similar SVM-based detection systems, they introduced a new way to extract features of obfuscation based on frequency. They realized the discrepancy on frequencies of text characters between obfuscation and non-obfuscation and consequently tried to measure such distinctions by directly comparing the observed frequencies, or utilized Shannon entropy to give a conclusion of such distinctions in overall. Their results showed that such frequency-based approach is a feasible way to conduct detection.

However, to straightly compare frequency of each character, one has to calculate and process over the frequency values of all 94 text characters of JavaScript, indicates the same amount of dimensions of feature vectors for describing data points, which also cannot achieve time and resource efficiency. In addition, this approach could be very sensitive to the observed frequencies and probably cause over-fitting. For instance, we need to collect samples of non-obfuscation as the training data to feed the classifier so that the classifier could know how to recognize the non-obfuscation. However, the observed frequencies of some rare text characters in non-obfuscated JavaScript may be strongly depend on the non-obfuscation samples collected which may not comprehensive enough to describe the general frequencies of these characters in non-obfuscation so that the over-fitting will occur. Such over-fittings will mislead the classifier to make the classification results become unreliable.

On the other hand, only relying on Shannon entropy is completely not enough.

C. Our improvement

In order to improve previous systems, we introduce new feature extraction approaches based on information theory, which each of these new features is more concentrated and have high robustness since these features capture the integral ‘statistical behaviors’ of a JavaScript but not depend on any trivial and facial ‘malicious’ feature.

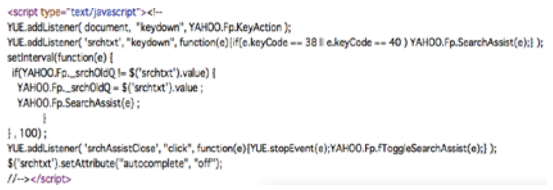
As a direct result, comparing to previous investigation, we reduce the amount of features to 7 and hence the operation of machine learning classifier will be conspicuously accelerated and will cost less system resources due to the reduction of dimensions of feature vectors. We achieved this goal while still keeping high detection rates.

On the other hand, nearly all of the previous works did not focus much on selection of machine learning classifiers and most of them utilized ordinary two–class SVM adjusted with “optimal” parameters gained from grid search. However, according to the customization of obfuscation programs and the obfuscation do not need to have any unified grammar restraint, even if obfuscation is distinct from most of non-obfuscation, each two unique obfuscation can be still very different so that assuming all or most of obfuscation belong to one single class and implement two-class SVM may not be suitable. Therefore it will be more precise to regard obfuscation as outliers compared to the non-obfuscation class. We will present our results to prove the correctness of this argument below. According to such condition, we propose a novelty detection approach which utilizing one-class SVM for detection, which fits this situation much better than two-class SVM. In addition, to train a one-class SVM, only samples of normal class (i.e. non-obfuscation) are needed, which solves the problem of unbalance data: it is easy to collect a large amount of non-obfuscated JavaScript samples by crawling over the Internet whereas the obfuscation samples are relatively very few due to their rare occurrences and very short life time. The unbalance data is also a problem that none of the previous investigations could solved.

III. Methodology

A. Classification of JavaScript

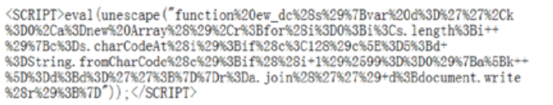
1) Non-obfuscation: JavaScript without any further process or only been minimized in order to decrease its length and accelerate webpage loading. Minimization always involves deleting the space characters; substitute long function names and so on. By a glimpse, compared to obfuscation, non-obfuscated JavaScript is much more similar to English texts.



```
<script type="text/javascript"><!--
YUE.addListener( document,  "keydown", YAHOO.Fp.KeyAction );
YUE.addListener( 'srchtxt', "keydown", function(e){if(e.keyCode == 38 || e.keyCode == 40 ) YAHOO.Fp.SearchAssist(e);} );
setInterval(function(e) {
 if(YAHOO.Fp._srchOldQ != $('srchtxt').value) {
  YAHOO.Fp._srchOldQ = $('srchtxt').value ;
  YAHOO.Fp.SearchAssist(e) ;
      }
} , 100) ;
YUE.addListener( 'srchAssistClose', "click", function(e){YUE.stopEvent(e);YAHOO.Fp.fToggleSearchAssist(e);} );
$('srchtxt').setAttribute("autocomplete", "off");
//--></script>
```

Fig. 1. An example of non-obfuscation JavaScript

2) Obfuscation: One may utilize varied algorithms to pack JavaScript by replacing the original code with other characters. The most significant feature of obfuscation is the existence of a heavily obfuscated payload contained in such script, which is a random combination formed by meaningless text characters.



```
<SCRIPT>eval(unescape("function%20ew_dc%28s%29%7Bvar%20d%3D%27%27%2Ck
%3D0%2Ca%3Dnew%20Array%28%29%2Cr%3Bfor%28i%3D0%3Bi%3Cs.length%3Bi++
%29%7Bc%3Ds.charCodeAt%28i%29%3Bif%28c%3C128%29c%5E%3D5%3Bd+
%3DString.fromCharCode%28c%29%3Bif%28%28i+1%29%2599%3D%3D0%29%7Ba%5Bk++
%5D%3Dd%3Bd%3D%27%27%3B%7D%7Dr%3Da.join%28%27%27%29+d%3Bdocument.write
%28r%29%3B%7D"));</SCRIPT>
```

Fig. 2. An example of obfuscation with an obfuscated ASCII encoding payload.

B. The characteristic of obfuscation

The most obvious feature of obfuscation compared to non-obfuscation is un-readable: an obfuscated payload does not need to obey any grammer rule of natural languages or the standard JavaScript as long as the payload could actually hide the maliciousness from the detection. The “un-readable” is mainly reflected in the abnormal frequencies of text characters in obfuscation and will cause the difference of observed frequencies of most characters between obfuscation and non-obfuscation. For instance, many obfuscated payloads usually contain many rare text charactors such as punctuator "%", "/" and upper case letters, which usually have very few appearances in the case of non-obfuscated JavaScript. In addition, as a special case of abnormal observed frequencies, many obfuscation programs work in a way of tautologically producing similar text strings to form the building blocks of the entire obfuscation payload so that such payloads always contain large amount of several specific repeated text characters (e.g. the punctuator "%" and number "2" in Fig.2), which results the observed frequencies of such characters significantly higher than others since the size of the obfuscation payloads always account for a very large proportion of the intergral JavaScript. Whereas in the case of non-obfuscation, the entire probability distribution of text charactors will be relatively close to uniform due to the restraints of English and programming grammar. We name this phenomenon as "repeated patterns". Logically, the appearance of "repeated patterns" will definitely giving the arising of the abnormal observed frequencies but not vice versa.

C. Support vector machine

1) Overview: SVM is a model in pattern recognition which is used for classification and regression around the data sets. Intuitively, for classification, SVM looks for a hyper-plane in feature space as the boundary to separate the data points.

2) One-class SVM: In this investigation, we implement one-class SVM: a classifier that is always utilized to detect novelty such as network intrusion. One-class SVM is a modification of ordinary two-class version, proposed by Scholkopf et al. in [9].

To train a one-class SVM, only normal samples are needed to form one normal class and any new input point will be either classified into this normal class and labelled as a normal point, or outside this class as an outlier. Intuitively, one maps a set of normal training data points into a feature space by a kernel function and initially regards the origin of the feature space as the only outlier. Then by using slack variables, one could separate the image of the normal class S, which is a class that includes most of the mapped training data points from the origin by a hyper-plane with maximum margin in order to estimate a discriminant function f which is positive on normal class S and negative on any point who is outside S. S is a small cluster with a simple geometric shape capturing most of the normal training data points.

For instance, assume $x_1, x_2, \ldots, x_n$ are training data points which all belong to the normal class S. $\Phi(x)$ is a set of basis functions that map the training data points to a feature space. To separate the training data from the origin, which is the original outlier, the following quadratic programming problem needs to be solved, which is very similar to the case of ordinary two-class SVM:

$$\min \frac{1}{2}\|w\|^2 + \frac{1}{vn}\sum_{i=1}^{n}\xi_i - \rho \qquad (1)$$

subject to $(w \cdot \Phi(x_i)) \geq \rho - \xi_i$ for $i=1,\ldots\ldots, n;\ \xi_i \geq 0$

The first term of the objective function penalizes complexity of the geometric shape of S while the second term penalizes the cost of errors caused when discriminant function f(x) is negative on the training data points of normal class by using slack variables ξ_i to allow some training data points lie on the wrong side of the hyper-plane as outliers. The parameter v represents an upper bound on the fraction of data that may be outliers. Solve this problem with respect to w and ρ could gain the definition of hyper-plane $w \bullet \Phi(x) - \rho = 0$ and the discriminant function $f(x) = \text{sign}((w \bullet \Phi(x)) - \rho)$.

D. Information-theoretic measures for extracting features.

For mining new features of obfuscation and non-obfuscation, we consider an application of measures from information theory so that a given JavaScript can be statistically characterized well by such measures. We then use discrepancies of these characteristics for classification between obfuscation and non-obfuscation. The calculation of these measures are based on unigram. Intuitively, we regard an input JavaScript as a text stream and assume characters of the stream are observed values generated from an identical random variable X, which takes values from 94 text characters of JavaScript.

The information-theoretic measures under consideration in this paper include several kinds of entropy and divergence (distance) (see [14] for various and important results in information theory) and they could be classified into two classes according to their motivations. Meanwhile, our approach also can be regarded as an extension of previous Unigram systems mentioned above. The first class includes entropy measures defined by one probability distribution, which are used to extract features of uncertainty of an input JavaScript. In previous investigations, Shannon entropy has been shown as an effective measure to detect obfuscation and we extend the entropy approach by introducing two more entropy measures: collision entropy and approximation of Shannon entropy. Theoretically, the entropy measures could only detect the abnormal uncertainty (i.e. repeated patterns) but can't be ensured to be able to discover the abnormal frequencies of text characters. For instance, to calculate Shannon entropy on the two text strings "setInterval" and "%20EW_DC%38" will give exactly the same entropy value but it is obvious that the latter one is more likely to be obfuscation as it contains many rare characters such as “%” and upper case letters. The second class includes distance measures: Kullback–Leibler divergence, Bhattacharyya distance and Euclidean distance, which are defined by two probability distributions. Be different from the entropy measures, the distance measures compare the observed frequencies of each text character respectively in obfuscation and non-obfuscation hence they can be used to detect any kinds of abnormal observed frequency phenomenon but not limited to “repeated patterns”. On the other hand, unlike previous Unigram systems which straightly comparing the frequencies of all 94 characters (i.e. 94-dimensional vectors), the distance measures allow us to conveniently and explicitly present the results of frequency comparisons between probability distributions in overall with scalar values. Overall, we can expect improvement of the previous results obtained by Shannon entropy and direct frequency comparison by using these new measures.

In addition, compare to previous Unigram systems, our method is relatively insensitive to the over-fitted observed frequencies. Even if there could be some over-fitting existing, since the calculation of each of our metric involves the frequencies of all text characters and give an overall evaluation with a single scalar value, hence the influence of few over-fitted observed frequencies can be mitigated.

1) Shannon entropy: In information theory, Shannon entropy measures the uncertainty of a random source (i.e. a probability distribution). Essentially, Shannon entropy is the expected value of the amount of “information”, where the term “information” is defined as the negative logarithm of the probability values. In existing investigations, Shannon entropy has been utilized to identify malicious randomness. However,nearly all of them didn’t give out a systematical depiction on its effectiveness of detecting obfuscation. In this investigation, we reused it as one of our measures, as well as comparing its effectiveness with other new measures we proposed.

According to our experiment, Shannon entropy is indeed helpful for detecting obfuscation as it is able to detect the appearances of repeated patterns that form the obfuscation payload. Intuitively, the several repeated text characters that are used to form the obfuscation payload will have significantly higher observed frequencies than others, which will result a lower Shannon entropy value compared to the case of non-obfuscation, whose observed frequencies of text characters are relatively close to uniform and will cause a higher Shannon entropy values.

For a random variable X, the Shannon entropy H(X) is defined by

$$H(X) = -\sum_x p(x)\log_2 p(x) \tag{2}$$

where the probability distribution p(x) is associated with X. In our system, p(x) is calculated by the observed frequency of a text character.

2) Kullback–Leibler(K-L) divergence: The K-L divergence D_{KL} (Q||P) or D_{KL} (P||Q) measures the difference between two probability distributions P and Q. In our experiments, we utilized it to measure if the probability distribution formed by observed frequencies of a given JavaScript is close to the “benign distribution”(see Section III-D), which is a probability distribution that describes the statistical feature of a standard non-obfuscation. If so, it implies that the given JavaScript has a similar frequencies of characters to the standard non-obfuscation JavaScript defined by “benign distribution” which could become one of the evidences that we could classify the given JavaScript as non-obfuscation. Since the K-L divergence is a non-symmetric measure, we calculate both D_{KL} (Q||P) and D_{KL} (P||Q) through the following definition:

$$D_{KL}(P\|Q) = \sum_x p(x)\log_2\frac{p(x)}{q(x)} \tag{3}$$

$$D_{KL}(Q\|P) = \sum_x q(x)\log_2\frac{q(x)}{p(x)}$$

where P and Q denote the “benign distribution” and the observed distribution of the given JavaScript respectively (usage of P and Q will be same below).

3) Approximation of Shannon entropy: We introduce an approach to approximate Shannon entropy based on asymptotic equipartition property (AEP). If $(x_1,x_2,\ldots\ldots,x_n)$ is an independent and identically distributed (i.i.d) sequence according to a probability distribution p(x), then we have:

$$\lim_{n\to\infty} -\frac{1}{n}\log_2 p(x_1,x_2,\ldots\ldots,x_n) \rightarrow H(X) \text{ in probability} \tag{4}$$

We next define the notion of the typical set: An i.i.d sequence $(x_1,x_2,\ldots\ldots,x_n)$will be included in the typical set $A_\varepsilon^{(n)}$ if its probability satisfies the following inequality:

$$H(X) - \varepsilon \le -\frac{1}{n}\log_2 p(x_1,x_2,\ldots\ldots,x_n) \le H(X)+\varepsilon \tag{5}$$

where ε is an arbitrarily small value. An important property of typical set includes that we have $\Pr\{A_\varepsilon^{(n)}\} > 1 - \varepsilon$, if n is sufficiently large.

We assume that the input JavaScript $(x_1,x_2,\ldots\ldots,x_n)$ is long enough and hence it belongs to the typical set. By assuming the input is non-obfuscation, we calculate the probability $p(x_1,x_2,\ldots\ldots,x_n)$ of the input JavaScript by using “benign distribution” (i.e. the value of each $p(x_i)$ is taken from the “benign distribution” where i =1,….,n) so that the probability value calculated indicates the chance of occurrence of the input JavaScript as a standard non-obfuscation. Then we evaluate $[-\frac{1}{n}\log p(x_1,x_2,\ldots\ldots,x_n)]$. By AEP, since we have assumed the input is a non-obfuscation, the value would approach to the Shannon entropy of the standard non-obfuscation, which can be calculated by "benign distribution". Otherwise the input is not a member of the typical set. However, based on the property of typical set mentioned above, the probability of an observed i.i.d sequence does not belong to typical set is negligible so that the input $(x_1, x_2,\ldots\ldots,x_n)$ then should be generated from another probability distribution which differs from "benign distribution" and belongs to its typical set. Therefore, we can determine if the value $[-\frac{1}{n}\log p(x_1,x_2,\ldots\ldots,x_n)]$ of the input approaches to the Shannon entropy of "benign distribution" by evaluating the absolute difference of these two values, to indirectly identify the similarity between the underlying probability distribution of the input JavaScript and "benign distribution". The input $(x_1, x_2,\ldots\ldots,x_n)$ is considered to be suspicious once its underlying distribution is conspicuously different from the "benign distribution".

Even if this approach is categorized as an uncertainty measure, it essentially evaluates the difference of observed frequencies between two JavaScript and behaves like a distance measure, so that unlike other uncertainty measures, it can be used to detect all kinds of abnormal observed frequencies but not only "repeated patterns". Compare to the distance measures such as K-L divergence, which straightforwardly compares the frequency values, the AEP approach examines if the times of appearances of each characters match the times it "should" have in a non-obfuscation, which indicates an indirect way to compare frequencies. Intuitively, for an input JavaScript, if most of the text characters have the "correct" times of appearances, then the value $[-\frac{1}{n}\log p(x_1,x_2,\ldots\ldots,x_n)]$ will actually approach to the Shannon entropy value of the "benign distribution".

4) Bhattacharyya distance: Similar to K-L divergence, the Bhattacharyya distance is also a measure for evaluating difference between two probability distributions P and Q over a finite set, it is defined by

$$D_B(P,Q) = -\ln\left(\sum_x \sqrt{p(x)q(x)}\right) \tag{6}$$

5) Collision entropy: The collision entropy is defined by

$$H_2(X) = -\log\sum_x p(x)^2 = -\log P(X=Y) \tag{7}$$

where random variables X and Y are given as independent and identically distributed according to a probability distribution p(x).

Except Shannon entropy, collision entropy is another kind of expression of the uncertainty of text characters. According to the characteristic of "repeated patterns" discussed above, the minority of text characters who have very high observed frequencies will cause a large collision probability P(X=Y) and results to a lower collision entropy, whereas in the case of non-obfuscation, the observed probability distribution would be relatively close to the uniform distribution which results in a higher collision entropy.

6) Euclidean distance: The Euclidean distance can be also utilized as a measure to evaluate difference between two probability distributions. It is defined by

$$d(P, Q) = d(Q, P) = \sqrt{\sum_x (q(x) - p(x))^2} \tag{8}$$

E. Benign distribution

We calculate several distance measures to inspect if the input JavaScript is statistically close to the standard non-obfuscation. By using samples from non-obfuscation training data set (See section IV-B) we collected, we introduce the "benign distribution" to model the standard non-obfuscation. Intuitively, "benign distribution" is an empirical probability distribution that describes the frequency of occurrence of each text character in our samples of non-obfuscation training data set. Each probability value p(x) of "benign distribution" is obtained by calculating the weighted average of frequency of each text character in JavaScript. For example, for a certain text character x, we calculate p(x) by the following formula

$$p(x) = \sum_n \frac{C}{T}\cdot \mathrm{freq}(x) \tag{9}$$

Where T denotes the total text length of all non-obfuscation samples within the data set; C denotes the length of a specific non-obfuscation sample in which the text character x occurs, freq(x) indicates the frequency of x observed in this non-obfuscation, and n counts the number of samples that x occurs.

IV. Experiments and Results

A. Sample data collection

We crawled the main pages of sites of Alexa "The top 500 sites" URL list[13] since December, 2014 and collected 2000 unique non-obfuscation samples. We also obtained 400 unique obfuscation samples from VirusTotal[11] and D3M 2010-2013 data sets[12]. Note that the obfuscation samples are not ensured to be malicious themselves but definitely suspicious as they were actually located within malicious HTML pages and heavily obfuscated. We discarded any JavaScript that less than 250 bytes, as their length is too short to reveal the frequency features. For all samples, we manually sieved to ensure there is no repeat.

B. *Constructing the classifier*

The one-class SVM model is built and trained through the LIBSVM package with R language, with a Radial basis function kernel and 10 cross validations. We randomly selected 300 non-obfuscation samples as training data (normal class) to train one-class SVM, and used the rest of non-obfuscation samples and all obfuscation samples as test data.

C. *Results of calculations:*

We compared calculated values of 7 measures we selected, around three data sets: Non-obfuscation training data, Non-obfuscation test data and Obfuscation test data. Fig.3 depicts a part of the comparisons.

Overall, we found that the values of measures of non-obfuscation samples in most cases are much stable and concentrated while the values of obfuscation are wild and random. Intuitively, most of the non-obfuscation samples are "similar" and gather within a cluster while the obfuscation samples are randomly located outside this cluster. Such results proved our assumption of obfuscation are outliers but not belong to one single class.

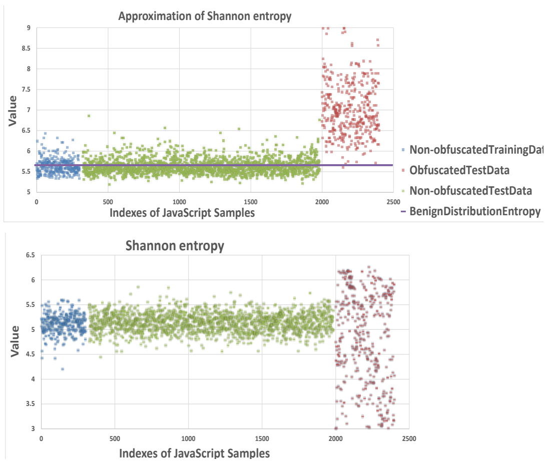


Fig. 3. Comparisons of values of Shannon entropy and approximation of Shannon entropy between three data sets

We also visualized the original 7-dimensional data in 3-dimensional space through Classical Multidimensional Scaling. Multidimensional scaling (MDS) is an approach to visualize the similarity of a set of high dimensional points. Of particularly, the classical MDS relocates the high dimensional points into 2 or 3-dimensional space while the Euclidean distance between each two points in original space are preserved as well as possible. As can be seen in Fig.4, the locations of data points indicates a "Normal class Versus Outlier" scheme.

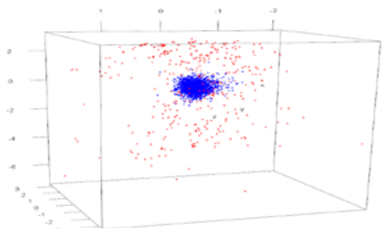


Fig. 4. Mapping original 7-dimensional data space into 3 dimensional coordinate by using classical MDS. Red plots indicate obfuscation while blue indicates non-obfuscation.

D. *Training and test results*

In order to test the effectiveness of each measure, we firstly trained the one-class SVM classifier by only utilizing each single measure to conduct the detection on three data sets and evaluated the accuracies. The results are shown in Fig.5.

As can be seen, each of our measures can individually be feed to a single classifier for detection where most of these classifiers gave good detection accuracies on non-obfuscation data while their results on obfuscation data set are middling. Namely, each of them is a weak classifier. From the figure, we could also compare the effectiveness of each measure. Especially, for detecting obfuscation, compare to Shannon entropy, which has been utilized within previous investigations, 4 of our new measures: K-L divergence D_{KL} (Q||P) and D_{KL} (P||Q), AEP entropy approximation and collision entropy averagely performed better, while the Euclidean distance has the similar performance to the Shannon entropy. It is also interesting to note that the statistical distance measures averagely have better performances compare to uncertainty measures, therefore it justified the fact foregoing mentioned above that distance measures could capture all kinds of abnormal observed frequencies whereas uncertainty measures are only able to detect the "repeated patterns", which is only a special case of the phenomenon of abnormal observed frequencies so that they are less effective compared to distance measures.

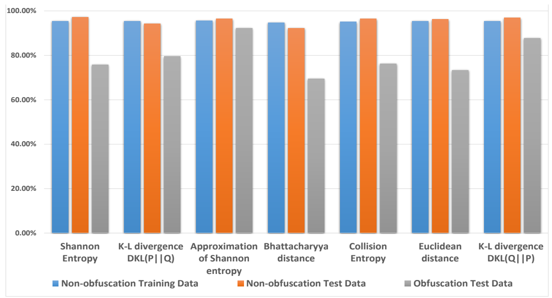


Fig. 5. Comparing effectiveness of measures

Then we combine these weak classifiers to one single strong classifier by feeding all of 7 features to a one-class SVM. The final detection accuracy of this combined system is shown in Table.1. According to the foregoing discussion, one may want to tune our system to minimize false-negative hence the accuracy tests are conducted under different conditions. The balance mode considers the tradeoff between false-negative and false-positive, while mode of minimizing false-negative favors false-negative as much as possible. Under balance mode, we respectively utilized all 7 features as well as only the top 4 effective features ranked by the results of Fig.5, which are K-L divergence D_{KL} (Q||P) and D_{KL} (P||Q), AEP entropy approximation and collision entropy, to conduct the test twice.

TABLE I. FINAL ACCURACY RESULTS ON THREE DATA SETS

Mode \ Data set	Obfuscation test data	Non-obfuscation test data	Non-obfuscation training data
Balance	96.20%	97.78%	97.01%
Balance (top 4 features)	95.50%	97.72%	97.67%
Minimizing false-negative	99.25%	81.42%	83.06%

E. Time consumption

We utilize all of our samples for testing if our system is time efficient. The test is conducted on a machine running Window7 Professional 64 bits, with Intel Xeon CPU E3-1225 CPU and 16GB RAM. We used build-in function “clock()” in C and “proc.time()” in R language to surround our feature extraction and classification programs to measure the time cost.

The test results are shown in Table.2. Note that the values are the total time to process the entire sample set but not time per sample. By evaluating the results, it is clear that our system is fast which is able to conduct large scale analysis.

TABLE II. TEST RESULTS OF TIME CONSUMPTION

Target of test	Feature extraction time	Classification time
2000 non-obfuscation samples	2.029 seconds	0.04 second
400 obfuscation samples	0.917 second	Less than 0.01 second

V. CONCLUSION AND FUTURE WORK

Our new filter system significantly reduced the dimensions of feature vectors while still giving high accuracies for classifying obfuscation and non-obfuscation. According to the fact that our measures do not count on specific and ambiguous ‘malicious’ behaviors, as well as the results of Fig.5 and Table.1, we showed that these measures have high robustness. We also justified the correctness of our modelling approach of novelty detection based on the results of data points distribution performed by Fig.3 and Fig.4, in which the non-obfuscation samples behave strong similarity and their values of measures always resemble while the obfuscation samples are randomly located outside the cluster of non-obfuscation. To sum it up, our systems performed high accuracies with practical time and resource, which can be operated over large scale smoothly.

On the other hand, as the detection of our system completely counts on the observed frequency and appearances of text characters of a given JavaScript, which requires the input JavaScript has to be long enough to reveal its real features of frequency. Therefore our system is not effective to inspect scripts that are too short and hence we made a threshold of minimum text length of 250 bytes in our experiment. Fortunately, short malicious obfuscation is rare.

Since our system only inspects obfuscation, one may have to combine our system to others to conduct a comprehensive detection of maliciousness, hence it will be necessary for us to test the compatibility of such combinations. On the other side, we will continue to test our system on other data sets as well as extension of our approach to detect other malicious codes.

ACKNOWLEDGEMENT

A part of this work was conducted under the auspices of the MEXT Program of Promoting the Reform of National Universities.

REFERENCES

[1] D. Canali, M. Cova, G. Vigna and C. Kruegel, “A Fast Filter for the Large-Scale Detection of Malicious Web Pages,” in Proc. of the 20th international conference on World wide web, 2011, pp.197–206.

[2] P. Likarish, EJ. Jung, I. Jo, “Obfuscated Malicious JavaScript Detection using Classification Techniques,”,in Proc. Of Malicious and Unwanted Software, 4th International Conference, 2009, pp. 47–53.

[3] W. Wang, Y. Lv, H. Chen, and Z. Fang, “A static malicious JavaScript detection using SVM,” in Proc. of the 2nd International Conference on Computer Science and Electronics Engineering, 2013.

[4] B. Kim, C. Im and H. Jung, “Suspicious malicious web site detection with strenth analysis of a JavaScript obfuscation,” International Journal of Advanced Science and Technology, Vol.26, January 2011

[5] M. Nishida et al., “Obfuscated malicious JavaScript detection using machine learning with character frequency,” Information processing society of Japan SIG Technical report, Vol. 2014.

[6] K. Rieck, T. Krueger, and A. Dewald, “Cujo: Efficient detection and prevention of drive-by download attacks,” in Proc. of the 26th Annual Computer Security Applications Conference, 2010, pp. 31-39.

[7] Y. Choi, T. Kim and S. Choi, “Automatic detection for JavaScript obfuscation attacks in web pages through string pattern analysis,” International Journal of Security and Its Applications, Vol. 4, No. 2, April, 2010, pp.13-26.

[8] L. M. Manevitz and M. Yousef, “One-Class SVMs for Document Classification,” Journal of Machine Learning Research, Vol. 2, 2002 pp. 139-154.

[9] B. Scholkopf, R. Williamson, A. Smola, J. Taylor and J. Platt, “Support Vector Method for Novelty Detection,” S.A. Solla, T.K. Leen and K.-R. Muller, pp. 582–588, MIT Press (2000).

[10] Y. Houa, Y. Changb, , T. Chenb, , C. Laihc, and C. Chena, “Malicious web content detection by machine learning”, Expert Systems with Application, January 2010, pp. 55–60.

[11] VirusTotal[online]. Available: https://www.virustotal.com/

[12] M. Kamizono et al, “Datasets for Anti-Malware Research - MWS Datasets 2013”, MWS2013, October, 2013

[13] Alexa Top Sites[online]. Available: http://www.alexa.com/topsites, 22 Dec, 2014

[14] T. M. Cover and T. A. Thomas, Elements of Information Theory, the 2nd edition, 2006, John Wiley and Sons, Inc.