Recommended Paper

Leaving All Proxy Server Logs to Paragraph Vector

Mamoru Mimura[1,a)] Hidema Tanaka[1,b)]

Received: January 17, 2018, Accepted: September 7, 2018

Abstract: Cyberattack techniques continue to evolve every day. Detecting unseen drive-by-download attacks or C&C traffic is a challenging task. Pattern-matching-based techniques and using malicious blacklists are not efficient anymore, because attackers easily change the traffic pattern or infrastructure to avoid detection. Therefore, many behavior-based detection methods have been proposed, which use the immutable characteristic of the traffic. These previous methods, however, focus on the attack technique, and can only detect drive-by-download (DbD) attacks or C&C traffic which have the immutable characteristic. These traditional methods have to devise the feature vectors. This paper proposes a generic detection method, which is independent of attack methods and does not need devising feature vectors. Our method uses Paragraph Vector, an unsupervised algorithm that learns fixed-length feature representations from variable-length texts and classifiers. Our method uses Paragraph Vector to capture the context in proxy server logs. We conducted cross-validation, timeline analysis and cross-dataset validation with multiple datasets. The experimental results show our method can detect unseen DbD attacks and C&C traffic in proxy server logs. The best F-measure achieved 0.97 in the timeline analysis and 0.96 on the other dataset.

Keywords: drive by download, C&C, Neural Network, Bag-of-Words, Word2vec, Paragraph Vector, Doc2vec, Support Vector Machine, Random Forests, Multi-Layer Perceptron

1. Introduction

Information and communication technology has been developing rapidly. New technology provides not only convenience but also new threats. Cyber threats are increasing drastically and public concern is at an all time high. Cyberattack techniques continue to evolve every day. Internet worms such as “Code Red” or “Blaster” used to be a main threat in the early 2000s. Since the second half of 2000s, Drive-by Download attack (DbD attack) and Spear Phishing attacks have been the main attack techniques on the Internet. In these attacks, attackers set malicious contents on websites or send email with malicious files to exploit vulnerabilities and run malicious code without the user’s knowledge. After the initial intrusion, the attacker takes control of the victim’s computer via a Command and Control (C&C) server over the Internet. The victim’s computer is used as a stepping stone to further deep drilling intrusion. In particular, a set of stealthy and continuous computer hacking processes is called APT (Advanced Persistent Threat). To detect cyberattacks, operators investigate IDS (Intrusion Detection System) alerts or logs recorded in network devices such as a firewall or a proxy server. In general, intrusion detection techniques on a network are classified roughly into methods using pattern matching and methods using blacklists. The methods using pattern matching are effective, if the malicious traffic contains a unique string pattern. IDS uses fixed strings or regular expression to describe their signatures. Malware used in APT attacks, however, communicates via a standard protocol, and attempts to imitate normal http communication (e.g., Plug X, Emdivi) [1], [2]. Therefore, it is difficult to describe the signatures. In this case, an IDS can use the malicious destination server (e.g., Landing site, Exploit site, C&C server) address as the signature. A firewall or a proxy server can also use the malicious destination server address as the blacklist. However, attackers can change the malicious destination servers easily to evade detection by network devices. In addition, the malicious server address has to be already-known before the cyberattack. Thus, detecting unseen malicious traffic is like a cat-and-mouse game so resolving this problem is a challenging task.

To tackle this challenging task, many behavior-based detection methods have been proposed. These methods capture the characteristics of DbD attacks or C&C traffic, and attempt to detect unseen malicious traffic. Many previous methods, however, focus on the attack technique. These methods can detect only DbD attacks (e.g., Refs. [2], [3], [4]) or C&C traffic (e.g., Refs. [7], [8]). Therefore, if the attack technique is changed then previous methods cannot respond. Besides security researchers have to devise the feature vectors to capture the characteristics. Furthermore, many previous methods require monitoring all network traffic. For instance, [4] requires web contents to detect DbD attacks. Countless organizations, however, do not keep all network traffic because of the size. Security incidents often occur in the organizations that did not take the countermeasures adequately. In most cases, there are inadequate log files to investigate the incident in the vulnerable organizations. Sometimes we might retrieve only log files on a single proxy server.

[1] National Defense Academy, Yokosuka, Kanagawa 239–8686, Japan
[a)] mim@nda.ac.jp
[b)] hidema@nda.ac.jp

The preliminary version of this paper was presented at Computer Security Symposium 2017 (CSS2017) in October 2017, and recommended to be submitted to Journal of Information Processing (JIP) by the program chair of CSS2017.

This paper focuses on the feature that Neural Network (NN) learns feature vector representation automatically. In recent years, NN has achieved remarkable results in the fields of image recognition, speech recognition or natural language processing (NLP). In the NLP fields, Word2vec [9] has proven outstanding. This model takes a large corpus of text and produces a vector space with each unique word in the corpus being assigned a corresponding vector in the space. This model enables not only document classification based on the appearance frequency, but also document classification based on the sense or the context. Furthermore, the same idea was extended to Paragraph Vector, an unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents. In this paper, we presume proxy server logs are written in a natural language, and attempt to learn the difference between benign traffic and malicious traffic automatically with Paragraph Vector. Then we input the extracted feature vectors with the label into supervised learning models to classify benign traffic and malicious traffic. Our generic detection method does not rely on attack techniques, and does not demand devising feature vectors.

The main contributions of this paper are three-fold:

(1) Propose a detection method using Doc2vec in proxy server logs [5].
(2) Verify that our method can detect DbD attacks in the long term [6].
(3) Verify that our method can detect C&C traffic in spear-phishing attacks.

The rest of the paper is organized as follows. The next section discusses related work and clarifies the difference among this method and previous methods. Section 3 presents Natural Language Processing (NLP) techniques. Section 4 proposes a generic detection method, which includes how to use NN and classifiers. Section 5 shows experimental results applying our method to datasets which contain DbD attacks and C&C traffic. Section 6 evaluates the method from a practical perspective and discusses the limitation.

2. Related Work

2.1 Behavior-based Detection

This paper aims to detect malicious traffic, even if both malicious server addresses and the distinctive traffic patterns are unknown. The malicious traffic includes DbD attacks and C&C traffic. In general, the main studies of network intrusion detection include signature-based detection and behavior-based detection. Signature-based detection relies on an existing signature database to detect known malicious traffic, and rarely detects unseen malicious traffic. Therefore, many behavior-based detection methods have been proposed. For example, some methods focused on the traffic classification from packet traces [10], [11], [12], [13]. However, analyzing packets is becoming intractable on broadband networks. The alternative approach is classification based on network logs such as DNS records, NetFlow or proxy server logs. There are several methods which use NetFlow [7], [8], [14] or DNS records [15], [16], [17], [18]. However, it is unusual to obtain all network traffic or logs in actual incidents. Therefore, we focus on proxy server logs.

2.2 Analyzing Proxy Server Logs

Kruegel et al. [19] categorized URIs by the path, and extracted the parameters from the query strings. Their statistical model learns the features, and detects the statistical outliers as attacks. This method expects detecting direct attacks to web servers such as buffer overflow, directory traversal, cross-site scripting and so on. Our method uses statistical machine learning models for binary classification of malicious traffic and benign traffic. Our method expects indirect attacks such as DbD attacks or SP attacks too.

Choi et al. [20] extracted feature vectors from the domain, path and so on included in URLs, and proposed a method using machine learning models to classify malicious URLs and benign URLs. This method uses not only proxy server logs but also the URL popularity, contents, DNS traffic or any other traffic. Our method uses only proxy server logs and does not require any other additional information obtained from external sources. Our method does not even require devising feature vectors.

Ma et al. [21] extracted feature vectors from the host name, top level domain name, path and so on included in URLs, and proposed an online learning algorithm to classify malicious URLs and benign URLs. This method divides URLs into tokens by the delimiter such as “dot” (.), “slash” (/), “question mark” (?), “equal” (=), “and” (&) and so on. Our method uses similar techniques to obtain tokens from URLs. These tokens serve as words to construct a corpus. Their method requires not only proxy server logs but also searching the whois database for IP address and domain name registration information, blacklists, the geographical feature, the bandwidth speed and so on. Our method uses only proxy server logs. Our method does not even require devising feature vectors.

Huang et al. [22] extracted feature vectors from the structure, the characteristic strings and the brand name included in URLs to detect malicious URLs with machine learning. This method aims to detect phishing URLs. Our method aims to detect DbD attacks and C&C traffic, however it is not limited to these attacks. Our method does not even require devising feature vectors.

Zhao et al. [23] focused on the cost to force users to analyze and label malicious traffic, and proposed an online active learning framework which updates the classifier to detect malicious URLs. This method uses the whois database for domain name registration information, blacklists and so on. Our method uses only proxy server logs and does not require any other information obtained externally.

Invernizzi et al. [24] built a network graph from IP addresses, domain names, FQDNs, URLs, paths, file names and so on. Their method focuses on the correlation among nodes to detect malware distribution. This method uses only the parameters obtained from proxy server logs. However, this method has to cover many IP address ranges, and performs in large-scale networks such as ISPs. In addition, this method needs the downloaded file types. Our method performs at any scale and does not need the downloaded file types.

Nelms et al. [25] focused on DbD attacks, and extracted the Lo-

cation field, Referrer field and so on from http Request messages and Response messages to build a URL transfer graph. They proposed a trace back system which could go back to the source from the URL transfer graph. This method uses hop counts, domain age and common features of the domain names to detect malicious URLs. Our method uses only proxy server logs and does not require any additional information obtained from external sources. In addition, our method can detect not only DbD attacks but also C&C traffic.

Bartos et al. [26] categorized proxy server logs into flows, and extracted many features from the URL, path, query, file name and so on. They proposed how to learn the feature vectors to classify malicious URLs. This method can decide the optimum feature vectors automatically. However, this method requires devising basic features for learning. Our method does not even require devising basic features.

Mimura et al. [1] categorized proxy server logs by FQDNs to extract feature vectors, and proposed a RAT (Remote Access Trojan or Remote Administration Tool) detection method using machine learning techniques. This method uses the characteristic that RATs continues to access the same path regularly. However, this method only works on C&C traffic. Our method can detect not only C&C traffic but also DbD attacks.

Shibahara et al. [2] focus on a sequence of URLs which include malicious artifacts of malicious redirections, and proposed a detection system which uses Convolutional Neural Networks. This method uses a honey client to collect URL sequences and their labels. However, this method performs for DbD attacks. Our method can detect not only DbD attacks but also C&C traffic.

3. Natural Language Processing (NLP) Technique

3.1 Bag-of-Words (BoW)

To calculate various measures to characterize a text, we have to transform the text into a vector. Bag-of-Words (BoW) is a simplifying representation used in NLP. In this model, a sentence is represented as the bag of its words, disregarding grammar and even word order but retaining multiplicity. BoW is commonly used in document classification methods where the frequency of each word is used as a feature for training a classifier. The most common type of features calculated from BoW is a term frequency, namely, the number of times a term appears in the sentence. However, term frequencies are not necessarily the best representation for the sentence. BoW cannot represent grammar, word order and word meaning. To represent unseen traffic, term frequencies are not efficient [5]. Furthermore, the feature dimension is defined by the vocabulary size of the data set. Therefore, we have to consider how to reduce the dimensionality in practical use.

3.2 Word2vec

Word2vec [9] is a model that produces word embedding. Word embedding is the collective name for a set of language modeling and feature learning techniques in NLP where words from the vocabulary are mapped to vectors of real numbers. This model is a shallow, two-layer neural network that is trained to reconstruct linguistic contexts of words. This model takes as its input a large corpus of text and produces a vector space, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to each other in the space. Word2vec is based on the distributional hypothesis, which motivates that the meaning of a word can be gauged by its context. Thus, if two words occur in the same position in two sentences, they are very much related either by semantics or syntax. Word2vec utilizes two algorithms to produce a distributed representation of words. One is Continuous-Bag-of-Words (CBoW), and the other is skip-gram. In the CBoW algorithm, the model predicts the current word from a window of surrounding context words. In the skip-gram algorithm, the model uses the current word to predict the surrounding window of context words. Word2vec allows calculating the semantic similarity between two words and infer similar words semantically. However, Word2vec is a model that merely produces word embedding. To calculate semantic similarity between two documents, this method has to be extended.

3.3 Paragraph Vector (Doc2vec)

An extension of Word2vec to construct embedding from entire documents has been proposed [27]. This extension is called Doc2vec or Paragraph2vec and has been implemented. Doc2vec is based on the same distributional hypothesis, which motivates that the meaning of a sentence can be gauged by its context. Thus, if two sentences occur in the same position in two paragraphs, they are very much related either in semantics or syntactic in the same way. Doc2vec utilizes two algorithms to produce Paragraph Vector a distributed representation of entire documents. One is Distributed-Memory (DM), and the other is Distributed-Bag-of-Words (DBoW). DM is an extension of CBoW, and the only change in this model is adding a document ID as a window of surrounding context words. DBoW is an extension of skip-gram, and the current word was replaced by the current document ID. Doc2vec allows calculating semantic similarity between two documents and infer similar documents semantically. Some implementations support also inference of document embedding on unseen documents. This function is important for developing a practical system to detect unseen malicious traffic. Because, unseen malicious traffic might include an unknown word (e.g., newly-changed FQDN, random strings).

4. Proposed Method

4.1 Purpose

The purpose of our method is detecting unseen malicious traffic, not filtering malicious URLs such as landing site, exploit site or C&C server addresses. Nowadays, attackers often use many compromised websites to imitate benign traffic. Hence, it is difficult to decide whether a URL is benign or malicious independently. To decide whether a URL is malicious or not, we have to investigate the context. We can investigate the context from sequential log lines in proxy logs. In this paper, the context is synonymous with sequential log lines. To investigate the context, we define malicious traffic as a paragraph which contains malicious URLs. The paragraph consists of 10 lines of proxy logs. As

a matter of course, a paragraph contains multiple URLs. Hence, our method detects malicious traffic which might contain benign URLs. We assume that these benign URLs are related to the malicious URLs. Our method does not extract only malicious URLs. Our method investigates a paragraph comprehensively and decides whether a paragraph is malicious or not. Our method expects that the operators perceive unseen malicious traffic based on the result from classifiers. Our method supports incident responders or digital forensic investigators.

4.2 Separating Logs with Spaces

This paper proposes an intrusion detection method in proxy server logs. The key idea of this research is processing proxy server logs as a natural language. In order to accomplish this, proxy server logs have to be separated into words. **Figure 1** shows a sample of proxy server logs.

This sample includes the date and time at which the transaction completed, request line from the client (includes the method, URL and user agent), HTTP status code returned to the client and size of the object returned to the client. Here, the 'client" signifies the user's computer which connects to servers over the Internet. A proxy server records the contents on a line in chronological order. The line originates the request from an internal client and is coupled with the response from the server.

Our method divides proxy server logs into HTTP status code (Status), request line from the client (Method and URL), size of the object returned to the client (size) and user agent (UA). Our method uses these 5 elements. Note that our method does not use source IP addresses. Furthermore, the request line is divided into method, URL and protocol version.

4.3 Separating URLs with Delimiters

We believe a URL is the most important element in detecting malicious traffic. Our method separates URLs with delimiters. **Figure 2** shows an example of a URL leaving a space between words.

First, our method divides a URL into scheme, FQDN (Fully Qualified Domain Name) and path which includes query strings. After that, our method separates the FQDN by "dot" (.). Then we can derive the top level domain name, sub domain name and so on, which means the country, organization, use or purpose (e.g., www, mail). Our method separates the path by "slash" (/) and "dot" (.), and also separates the query string by "question mark" (?), "equal" (=) and "and" (&). Then, we can derive the directory name, file name, extension from the path. We can also derive the variable names and values from the query string, which are used in the running program on the server. We decided to leave the delimiters as a word for constructing a corpus. Because the delimiters are related to the contiguous word meanings. For instance, some delimiters such as "slash" (/) are closely related to the structure of the URL.

To summarize, our method divides URLs into words by the delimiters which are "dot" (.), "slash" (/), "question mark" (?), "equal" (=) and "and" (&). These words construct a corpus to train a language model.

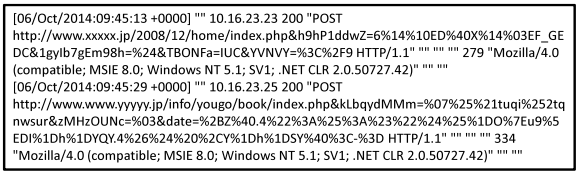

Fig. 1 A sample of proxy server logs.

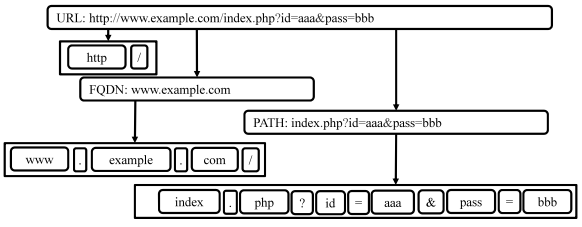

Fig. 2 An example of a URL leaving a space between words.

4.4 Overview

Figure 3 shows an overview of the proposed method. First, our method constructs a corpus from malicious proxy server logs and benign proxy server logs. Both logs are separated by the previously mentioned method. Then, our method constructs a vector space from the corpus, and converts both proxy server logs into vectors with the labels. Our method interprets 10 lines as a paragraph. This parameter was decided based on an empirical approach [5]. The language model which constructs a vector space is Doc2vec. These labeled vectors are training data for classifiers. The classifiers are Support Vector Machine (SVM), Random Forests (RF) and Multi-Layer Perceptron (MLP).

A SVM model is a representation of the training data as points in space, mapped so that the training data of the separate categories are divided by a clear gap that is as wide as possible. Test data are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. RF are an ensemble learning method that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees. MLP is a class of feedforward artificial neural

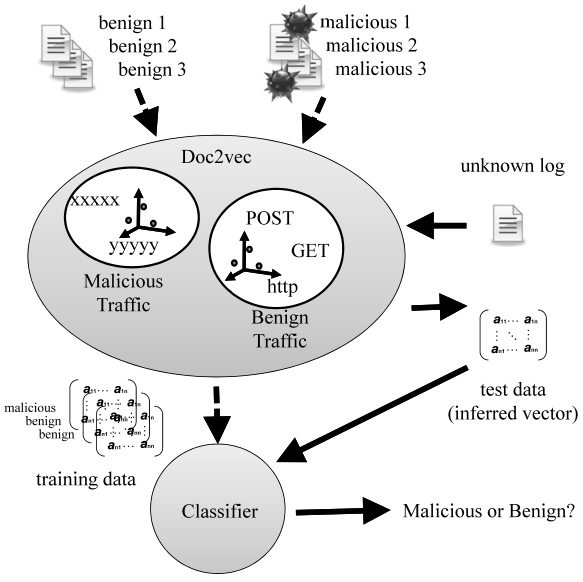

Fig. 3 An overview of the proposed method.

network, which consists of at least three layers of nodes. Each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training.

These are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training data, each labeled as belonging to one or the other of two categories, these training algorithms build a model that assigns new examples to one category or the other. After that, we convert unseen proxy server logs into vectors. These unlabeled vectors are test data for the classifiers. Finally, we input these unlabeled vectors to the classifiers, and can obtain a predicted label. The predicted label is either malicious or benign.

4.5 Implementation

Our method was developed by Python-2.7 with open source machine learning libraries, gensim-1.01 [28], scikit-learn-0.18.0 [29] and chainer-1.23 [30].

Gensim is a Python library that provides unsupervised semantic modelling from plain text, and includes a BoW model and a Doc2vec model. **Table 1** shows the parameters for the Doc2vec model. We set the dimensionality of the feature vectors 100, and chose DBoW which was an extension of skip-gram. The window is the maximum distance between the predicted word and context words used for prediction within a document. The minimum count is the threshold value to ignore all words with a total frequency lower than this.

Scikit-learn is a machine-learning library for Python that provides tools for data mining with a focus on machine learning, and supports SVM and RF. Our method uses a SVC function with a linear kernel for SVM. Our method also uses a RandomForestClassifier function for RF.

Chainer is a flexible Python framework for neural networks, which supports MLP with CUDA computation. We use CUDA 8.0 and cuDNN-6.0. **Table 2** shows the parameters for the MLP model. The number of input layer units is the dimensionality of the test data. Thus, the dimensionality is 100 as we mentioned before. The number of labels is 2, namely benign or malicious. ReLU (Rectified Linear Unit) is an activation function defined as follows.

$$f(x) = x^+ = \max(0, x)$$

It is also known as a ramp function and has been used in convolutional networks more effectively than the widely used logistic sigmoid. Adam (Adaptive moment estimation) is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments [31]. This method is well suited for problems that are enormous in terms of data and parameters, and also appropriate for non-stationary objectives and problems with very noisy and sparse gradients. We use cross entropy to define the loss function in optimization.

Table 1 The parameters for the Doc2vec model.

Dimensionality of the feature vectors	100
Window	15
Minimum count	2
Number of epochs	30
Training algorithm	DBoW

Table 2 The parameters for the MLP model.

Number of input layer units	100
Number of hidden layer units	500
Number of labels	2
Activation function	ReLU
Dropout ratio	0
Minibatch size	100
Optimizer	Adam

5. Experiment

5.1 Dataset

To evaluate our method, we use captured pcap files from Exploit Kit (EK) between 2014 and 2017, which were downloaded from the website MALWARE-TRAFFIC-ANALYSIS.NET [32]. EK is a software kit designed to run on web servers, with the purpose of identifying software vulnerabilities in client machines communicating with it. EK discovers and exploits vulnerabilities that might allow uploading and running malicious code on the client computer. We chose some EKs which communicate via a standard protocol and attempt to imitate normal http communication. We name these pcap files MTA dataset. We also use BOS (Behavior Observable System), D3M (Drive-by Download Data by Marionette) dataset and NCD (Normal Communication Data in MWSCup 2014). These datasets are parts of the MWS datasets [33], and include pcap files. BOS and D3M contain malicious traffic. **Table 3** shows the details.

These pcap files are malicious. However, the property is different. MTA is a set of packet traces downloaded from the website. This dataset includes traffic from many EKs (e.g., Angler EK, Neutrino EK, Magnitude EK, RIG EK, Nuclear EK). This dataset contains the traffic from the latest version of EKs. We use this dataset to verify that our method is effective against DbD attacks. BOS comprises some cases of malware samples (e.g., Emdivi, Plug X), packet traces, and a process log collected from a virtual company that is a malware execution environment. The malware that were attached to e-mails are provided as well as the observed attackers activities on the host within the virtual company. BOS contains C&C traffic which uses a standard protocol and attempts to imitate normal http communication. Thus,

Table 3 The detail of the original datasets.

Period	Size			
	MTA	BOS	D3M	NCD
2010	-	-	130M	-
2011	-	-	24.8M	-
2012	-	-	33.2M	-
2013	-	-	14.6M	-
2014	238M	38.9M	22.3M	6.4G
2015	186M	2.25G	334M	-
2016	373M	3.48G	-	-
2017	109M		-	-

Table 4 The detail of the converted datasets.

		MTA	BOS	D3M	NCD
Size		3.9M	3.3M	2.4M	8.3M
Line		5,578	6,121	9,611	25,151
Status	unique	20	8	14	17
	frequent	3,580	3,840	5,121	17,754
Method	unique	3	2	4	4
	frequent	4,734	3,474	6,992	24,657
URL	unique	4,049	2,655	4,216	18,023
	frequent	32	2,195	38	117
Size	unique	1,060	193	1,073	1,320
	frequent	140	1,712	368	2,840
UA	unique	95	16	15	238
	frequent	927	2,279	6,316	2,840

this traffic is difficult to distinguish from benign traffic. We use this dataset to verify that our method can detect C&C traffic in spear-phishing attacks. D3M is a set of packet traces collected from the web-client, high-interaction honeypot system, which is based on Internet Explorer on Windows OS with several vulnerable plugins, such as Adobe Reader, Flash Player, Java and so on. Though this dataset is related with EKs (e.g., Blackhole Exploit Kit, Elenore, Mpack), does not include the latest version of EKs. Thus, we use this dataset supplementally. NCD is a benign pcap file which was captured during a day in 2014.

Our method aims to detect malicious traffic from proxy server logs. Thus, we have to convert these pcap files into pseudo proxy server logs. We extracted http traffic from these pcap files, and coupled the requests and responses. **Table 4** shows the details.

This table includes the size, number of lines, unique number of each element and number of the most frequent value.

Then, we compounded the malicious logs and the benign logs into datasets. We split the datasets into training data and test data to conduct 10-fold cross-validation, timeline analysis and cross-dataset validation. In the 10-fold cross-validation, we use all the datasets. In the timeline analysis and cross-dataset validation, we switch malicious datasets for the training data and test data. We use the first half of the NCD dataset as the training data and the rest half as the test data. Our method uses only training data to construct a corpus. Because, we presume that the test data is unseen traffic.

5.2 Metrics

Three evaluation metrics are used: precision (P), recall (R) and f-measure (F). These metrics are used to evaluate the performance of classification tasks.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F - measure = \frac{2Recall \times Precision}{Recall + Precision}$$

TP (True Positive), *FP* (False Positive), *TN* (True Negative) and *FN* (False Negative) are defined as shown in **Table 5**.

TP is the number of instances correctly classified as benign or malicious, *TN* is the number of instances correctly classified as Not-benign or Not-malicious. *FP* is the number of instances incorrectly classified as benign or malicious, *FN* is the number of instances incorrectly classified as Not-benign or Not-malicious. In this paper, malicious traffic is a paragraph which contains malicious URLs. Our method interprets 10 lines as a paragraph and adds the label to the paragraph. Hence, we calculate TP and FP from the number of classified paragraphs.

Table 5 Confusion matrix for two possible outcomes.

		True label	
		Positive	Negative
Predicted label	Positive	*TP*	*FP*
	Negative	*FN*	*TN*

Table 6 Experiment environment.

CPU	Core i7-3770 3.4 GHz
Memory	DDR4 SDRAM 16 GB
GPU	GeForce GTX980/4G
OS	Windows 7

Table 7 The result of the 10-fold cross-validation.

dataset	classifier	Benign			Malicious		
		P	R	F	P	R	F
MTA (DbD)	SVM	0.99	0.98	0.98	0.95	0.98	0.97
	RF	0.96	1.00	0.98	0.99	0.89	0.94
	MLP	0.99	1.00	0.99	0.99	0.97	0.98
BOS (C&C)	SVM	1.00	1.00	1.00	1.00	1.00	1.00
	RF	0.98	1.00	0.99	1.00	0.93	0.96
	MLP	1.00	1.00	1.00	1.00	0.99	0.99

5.3 Experimental Environment

Table 6 shows the experimental environment. This method needs only a simple computer, and does not require special equipment.

5.4 Result

Table 7 shows the results of the 10-fold cross-validation. RF is less accurate than other classifiers.

Table 8 shows the results of the timeline analysis (MTA). All the run times for each classification were less than 0.1 second. SVM achieved a good accuracy on average. The other classifiers were less accurate and unstable than SVM. Focusing on the same test data, new training data did not always improve the accuracy.

Table 9 shows the results of the timeline analysis (BOS). All the run times for each classification were less than 0.1 second. In C&C traffic, SVM achieved the best accuracy.

Table 10 shows the results of the cross-dataset validation. All the run times for each classification were less than 0.1 second. SVM and MLP achieved a good accuracy on average. RF was less accurate than the other classifiers.

As a result, SVM achieved a good accuracy on average. RF and MLP could not derive stable results. A possible reason for this is that Doc2vec generates linearly separable vectors. Another possible reason is over fitting. Hence, we concluded the best classifier was SVM.

Table 8 The result of the timeline analysis (MTA).

training data	test data	classifier	Benign			Malicious		
			P	R	F	P	R	F
2014	2015	SVM	1.00	0.96	0.98	0.84	1.00	0.91
		RF	0.98	0.98	0.98	0.91	0.87	0.89
		MLP	1.00	0.98	0.99	0.89	0.98	0.93
2014	2016	SVM	1.00	0.97	0.99	0.86	1.00	0.92
		RF	0.98	0.99	0.99	0.96	0.90	0.93
		MLP	0.94	0.98	0.96	0.93	0.78	0.85
2014	2017	SVM	1.00	0.97	0.98	0.85	1.00	0.97
		RF	0.98	0.99	0.99	0.96	0.91	0.93
		MLP	0.99	0.98	0.98	0.60	0.79	0.68
2015	2016	SVM	1.00	0.98	0.99	0.93	1.00	0.96
		RF	0.91	1.00	0.95	0.99	0.66	0.79
		MLP	0.98	1.00	0.99	0.86	0.66	0.75
2015	2017	SVM	1.00	0.98	0.99	0.69	0.94	0.80
		RF	0.98	1.00	0.99	1.00	0.64	0.78
		MLP	0.87	0.88	0.87	0.89	0.88	0.89
2016	2017	SVM	1.00	0.99	0.99	0.74	0.94	0.83
		RF	0.99	1.00	0.99	0.92	0.69	0.79
		MLP	0.99	1.00	1.00	0.96	0.88	0.92

Table 9 The result of the timeline analysis (BOS).

training data	test data	classifier	Benign			Malicious		
			P	R	F	P	R	F
First Half	Rest Half	SVM	0.94	1.00	0.97	0.99	0.76	0.86
		RF	0.90	1.00	0.94	1.00	0.55	0.71
		MLP	0.92	1.00	0.96	1.00	0.62	0.76

Table 10 The result of the cross-dataset validation.

training data	test data	classifier	Benign			Malicious		
			P	R	F	P	R	F
MTA	D3M	SVM	0.97	0.91	0.94	0.92	0.97	0.94
		RF	0.89	0.98	0.93	0.96	0.81	0.88
		MLP	0.96	0.98	0.97	0.96	0.93	0.94
D3M	MTA	SVM	0.97	0.97	0.97	0.96	0.96	0.96
		RF	0.90	0.99	0.94	0.98	0.86	0.92
		MLP	0.93	0.98	0.95	0.97	0.91	0.94

6. Discussion

6.1 Accuracy

Results from the experiments demonstrate that our method is effective. This is because there are some differences in words and the structure. The proposed method divides URLs into words by the delimiters, and Doc2vec illustrates the ratio of the co-occurrence probabilities of the two words within the window size. Some delimiters are closely related to the structure of the URL. The proposed method learns the structure automatically, even if a human makes no clear indication.

There are some causes of the false-positives and false-negatives. The primary cause was the sites which provided web APIs (e.g., Authentication, Streaming). A web API is an application programming interface (API) for either a web server or a web browser. A web API produces interactive communication with numerous parameters to provide a coating high functionality and convenience. This behavior is similar to Exploit Kits or C&C traffic. The attacker needs numerous parameters to control the victim's computer at will. Thus, Exploit Kits and C&C traffic have to produce interactive communication with numerous parameters. The other cause was the sites which provided update programs or pattern files (e.g., Anti Virus Scanner). This behavior is similar to downloading malware and malware infections. We can mitigate these false positives with the whitelist. Besides, some benign URLs were mixed in the malicious traffic. In a sense, these impurities are not the cause. We can enhance purity of the malicious traffic to improve the accuracy.

6.2 Adaptability

Our method can detect DbD attacks and C&C traffic as malicious traffic by the same method. All we have to do is input malicious and benign proxy server logs. Our method can detect malicious traffic regardless of the attack technique. No prior knowledge of the attack techniques is required for capturing the characteristic. Our method does not use different detection techniques. If attackers change their attack techniques, our methods learn the characteristic automatically. Besides our method does not require setting the feature vectors. Hence, our method is adaptable to many attack techniques.

6.3 Durability

Our method learns the difference between benign traffic and malicious traffic automatically with neural networks. In neural networks, it is difficult to specify what feature of an input data a specific feature map captures. This means that an attacker cannot recognize the features either. We tried some experiments to specify the feature. We conducted the same experiments without some elements (e.g., FQDN, User Argent). However, there was no notable change in the performance. This might mean that our method does not rely on a specific element. In that case, an attacker has no effective countermeasure to evade this method. The only option is imitating normal communication completely. Thus, our method is effective and durable in the long term.

6.4 Required Time

Our method requires only few minutes to construct the language model and the classifier. In this experiment, our method took roughly three minutes on average. The greater the pcap files or the log files to construct the models, the longer the required time. However, we can construct the models in advance. In this paper, our method could classify unseen logs with these pre-trained models within a second. Thus, our method can analyze network traffic or proxy server logs in real time.

6.5 Practical Use

The proposed method was effective on the other dataset. This means the proposed method is powerful and versatile. Many previous methods require monitoring all network traffic. Many organizations, however, do not retain network traffic. Thus, these methods are not practical. The proposed method does not require monitoring all network traffic, and only uses malicious and benign proxy server logs.

In this paper, we obtained pseudo proxy server logs from

pcap files. We can obtain these malicious pcap files easily from websites such as MALWARE-TRAFFIC-ANALYSIS.NET, which disclose malicious traffic data. We can also obtain benign pcap files or actual proxy server logs easily from everywhere. Our method therefore has few constraints on practical use.

6.6 Ethics

In this paper, we used malicious pcap files and benign pcap files. These files might contain privacy sensitive information such as personal information, email addresses and client's IP addresses. Many previous methods require monitoring all network traffic. Therefore, the possibility of accessing the payloads cannot be denied. The payloads might contain personal information and email addresses. However, our method does not require monitoring all network traffic. Moreover, our method does not require client's IP addresses, and does not even require identifying the client's sources.

In practical use, our method uses only pre-trained models to detect malicious traffic. The models do not include any payload and logs. Therefore, we can share or disclose the pre-trained models without much resistance.

7. Conclusion

In this paper, we describe how to construct a corpus from proxy server logs and apply Doc2vec to learn the difference of benign traffic and malicious traffic automatically. Then, we propose the generic detection method using supervised learning models to classify benign traffic and malicious traffic. This paper conducts cross-validation and timeline analysis with MTA dataset and BOS dataset. This paper also demonstrates cross-dataset validation with MTA dataset and D3M dataset. Consequently, the proposed method can detect both unseen DbD attacks and C&C traffic in proxy server logs. We verify that the proposed method is effective over three years, and effective on the other dataset too. The best F-measure achieves 0.97 in the timeline analysis and 0.96 on the other dataset.

Applying our method to actual proxy server logs is a future work. In this paper, we used the datasets which contains DbD attacks, C&C traffic and benign traffic. A large network has many clients which access various websites. While load balancers can maintain a stable state for the duration of a client's session, our method might have to be improved. One improvement plan is adjusting the size of a paragraph. This might integrate consecutive lines originated from each client into a paragraph. Another plan is using other NLP techniques to summarize a paragraph. These techniques extract important words from various words in proxy server logs. We should evaluate the accuracy in a practical condition. Another future work is how to update the model. In this experiment, new training data did not always improve the accuracy. In this time, we presumed proxy server logs were written in a natural language. We can presume any other logs such as IDS alerts, firewall logs, SIEM (Security Information and Event Management) events are written in a natural language in the same manner. We believe this would allow classifying the details automatically.

References

[1] Mimura, M., Otsubo, Y., Tanaka, H. and Tanaka, H.: A Practical Experiment of the HTTP-Based RAT Detection Method in Proxy Server Logs, *Proc. 12th Asia Joint Conference on Information Security*, pp.31–37 (2017).

[2] Shibahara, T., Yamanishi, K., Takata, Y., Chiba, D., Akiyama, M., Yagi, T., Ohsita, Y. and Murata, M.: Malicious URL Sequence Detection using Event De-noising Convolutional Neural Network, *Proc. IEEE ICC 2017 Communication and Information Systems Security Symposium* (2017).

[3] Takata, Y., Akiyama, M., Yagi, Y., Hariu, T. and Goto, G.: MineSpider: Extracting URLs from Environment-Dependent Drive-by Download Attack, *Proc. 2015 IEEE 39th Annual Computer Software and Applications Conference*, Vol.2, pp.444–449 (2015).

[4] Jodavi, M., Abadi, M. and Parhizkar, E.: DbDHunter: An ensemble-based anomaly detection approach to detect drive-by download attacks, *Proc. 2015 5th International Conference on Computer and Knowledge Engineering* (*ICCKE*), pp.273–278 (2015).

[5] Mimura, M. and Tanaka, H.: Heavy Log Reader: Learning the Context of Cyber Attacks Automatically with Paragraph Vector, *Proc. 13th International Conference on Information Systems Security*, *LNCS*, Vol.10717, pp.146–163 (2017).

[6] Mimura, M. and Tanaka, H.: Long-term Performance of a Generic Intrusion Detection Method Using Doc2vec, *Proc. 4th International Workshop on Information and Communication Security*, pp.456–462 (2017).

[7] Gu, G., Perdisci, R., Zhang, J. and Lee, W.: Botminer: Clustering Analysis of Network Traffic for Protocol and Structure Independent Botnet Detection, *Proc. USENIX Security Symposium*, Vol.5, pp.139–154 (2008).

[8] Bilge, L., Balzarotti, D., Robertson, W., Kirda, E. and Kruegel, C.: Disclosure: Detecting Botnet Command and Control Servers through Large-scale Netflow Analysis, *Proc. 28th Annual Computer Security Applications Conference*, pp.129–138 (2012).

[9] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J.: Distributed Representations of Words and Phrases and Their Compositionality, *Advances in Neural Information Processing Systems*, pp.3111–3119 (2013).

[10] Wang, K. and Stolfo, S.: Anomalous Payload-based Network Intrusion Detection, *LNCS*, Vol.3224, pp.203–222 (2004).

[11] Moore, D., Shannon, C., Brown, D.J., Voelker, G.M. and Savage, S.: Inferring Internet Denial-of-service Activity, *ACM Trans. Computer Systems*, Vol.24, No.2, pp.115–139 (2006).

[12] Bailey, M., Oberheide, J., Andersen, J., Mao, Z., Jahanian, F. and Nazario, J.: Automated Classification and Analysis of Internet Malware, *LNCS*, Vol.4637, pp.178–197 (2007).

[13] Song, H. and Turner, J.: Toward Advocacy-free Evaluation of Packet Classification Algorithms, *IEEE Trans. Computers*, Vol.60, No.5, pp.723–733 (2011).

[14] Karagiannis, T., Papagiannaki, K. and Faloutsos, M.: Blinc: Multilevel Traffic Classification in the Dark, *Proc. 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp.229–240 (2005).

[15] Antonakakis, M., Perdisci, R., Dagon, D., Lee, W. and Feamster, N.: Building a Dynamic Reputation System for DNS, *Proc. 19th USENIX Security Symposium* (2010).

[16] Antonakakis, M., Perdisci, R., Lee, W., Vasiloglou II, N. and Dagon, D.: Detecting Malware Domains at the Upper DNS Hierarchy, *Proc. 20th USENIX Security Symposium* (2011).

[17] Antonakakis, M., Perdisci, R., Nadji, Y., Vasiloglou, N., Abu-Nimeh, S., Lee, W. and Dagon, D.: From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware, *Proc. 21th USENIX Security Symposium* (2012).

[18] Rahbarinia, B, Perdisci, R. and Antonakakis, M.: Segugio: Efficient Behavior-Based Tracking of New Malware-Control Domains in Large ISP Networks, *Proc. 2015 IEEE/IFIP International Conference on Dependable Systems and Networks* (2015).

[19] Kruegel, C. and Vigna, G.: Anomaly Detection of Webbased Attacks, *Proc. 10th ACM Conference on Computer and Communications Security*, pp.251–261 (2003).

[20] Choi, H., Zhu, B.B. and Lee, H.: Detecting Malicious Web Links and Identifying Their Attack Types, *Proc. 2nd USENIX Conference on Web Application Development*, pp.1–11 (2011).

[21] Ma, J., Saul, L.K., Savage, S. and Voelker, G.M.: Learning to Detect Malicious URLs, *ACM Trans. on Intelligent Systems and Technology*, Vol.2, No.3, Article 30 (2011).

[22] Huang, H., Qian, L. and Wang, Y.: A SVM-based Technique to Detect Phishing URLs, *Information Technology Journal*, Vol.11, No.7, pp.921–925 (2012).

[23] Zhao, P. and Hoi, S.C.: Cost-sensitive Online Active Learning with Application to Malicious URL Detection, *Proc. 19th ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining, pp.919–927 (2013).
[24] Invernizzi, L., Miskovic, S., Torres, R., Saha, S., Lee, S., Mellia, M., Kruegel, C. and Vigna, G.: Nazca: Detecting Malware Distribution in Large-scale Networks, *Proc. Network and Distributed System Security Symposium* (2014).
[25] Nelms, T., Perdisci, R., Antonakakis, M. and Ahamad, M.: Webwitness: Investigating, Categorizing, and Mitigating Malware Download Paths, *Proc. 24th USENIX Security Symposium*, pp.1025–1040 (2015).
[26] Bartos, K. and Sofka, M.: Optimized Invariant Representation of Network Traffic for Detecting Unseen Malware Variants, *Proc. 25th USENIX Security Symposium*, pp.806–822 (2016).
[27] Le, Q. and Mikolov, T.: Distributed Representations of Sentences and Documents, *Proc. 31st International Conference on Machine Learning*, pp.1188–1196 (2014).
[28] gensim, available from ⟨https://radimrehurek.com/gensim/⟩.
[29] scikit-learn, available from ⟨ http://scikit-learn.org/⟩.
[30] Chainer, available from ⟨https://chainer.org/⟩.
[31] Kingma, D.P. and Ba, J.: Adam: A Method for Stochastic Optimization, *Proc. 3rd International Conference for Learning Representations* (2015).
[32] MALWARE-TRAFFIC-ANALYSIS.NET, available from ⟨http://www.malware-traffic-analysis.net/⟩.
[33] Hatada, M., Akiyama, M., Matsuki, T. and Kasama, T.: Empowering Anti-malware Research in Japan by Sharing the MWS Datasets, *Journal of Information Processing*, Vol.23, No.5, pp.579–588 (2015).

Editor's Recommendation

Authors propose a generic detection method that uses Paragraph Vector to capture the context in proxy server logs. This method can detect unseen DbD attacks and C&C traffic in proxy server logs. In addition, this method can detect malicious traffic with realistic calculation cost and log volume. Therefore, we expect the proposed method can be used widely and many readers will be interested.

(Program Chair of Computer Security Symposium 2017 (CSS2017), Yuji Suga)

Mamoru Mimura received his B.E. and M.E. in Engineering from National Defense Academy of Japan, in 2001 and 2008 respectively. He received his Ph.D. in Informatics from the Institute of Information Security in 2011 and M.B.A. from Hosei University in 2014. During 2001–2017, he was a member of the Japanese Maritime Self Defense Forces. During 2011–2013, he was with the National Information Security Center. Since 2014, he has been a researcher in the Institute of Information Security. Since 2015, he has been with the National center of Incident readiness and Strategy for Cybersecurity. Currently, he is an Associate Professor in the Department of C.S., National Defense Academy of Japan.

Hidema Tanaka received his B.E., M.E., and Ph.D. all in Electrical Engineering from Science University of Tokyo, in 1995, 1997 and 2000 respectively. He was a Director of Security Fundamentals Laboratory at the National Institute of Information and Communications Technology until 2011. Currently, he is an Associate Professor in the Department of C.S., National Defense Academy of Japan.