Vol.2019-DPS-178 No.16
Vol.2019-CSEC-84 No.16
2019/3/4

A Linguistic Approach to Detect Exploit Kits in Actual Proxy Logs

Mamoru Mimura[1,a)]

概要 : Modern http-based malware imitates benign traffic to evade detection. To detect unseen malicious traffic, a linguistic-based detection method for proxy logs has been proposed. This method extracts words as feature vectors automatically with natural language techniques, and discriminates between benign traffic and malicious traffic. The previous method generates a corpus from the whole extracted words which contain trivial words. To generate discriminative feature representation, a corpus has to be effectively summarized. In actual proxy logs, benign traffic is dominant, and occupies malicious feature representation. Therefore, the previous method does not perform accuracy in practical environment. This paper demonstrates that the previous method is not effective in actual proxy logs because of the imbalance. To mitigate the imbalance, our method extracts important words from proxy logs based on the frequency. We performed cross-validation and timeline analysis with captured pcap files from Exploit Kit and actual proxy logs. The experimental results show our method can detect unseen malicious traffic in actual proxy logs. The best F-measure achieves 0.95 in the timeline analysis.

1. Introduction

Modern malware imitates benign traffic to evade detection, and detecting unseen malicious traffic is a challenging task. To detect unseen malicious traffic, many methods using machine learning techniques have been proposed. Some methods, however, require packet traces[5], or additional parameters[6]. Thus, most previous methods are not applicable to actual information systems.

To tackle a realistic threat, a linguistic-based detection method for proxy logs has been proposed[1]. This method divides proxy logs into words and uses the words as feature vectors. To convert the words into feature vectors automatically, this method utilizes Paragraph Vector a natural language technique. In this method, Paragraph Vector provides feature vectors to discriminate between benign traffic and malicious traffic. To generate a discriminative feature vectors, an appropriate corpus is required. The previous method generates a corpus from the whole extracted words which contain trivial words. Since these trivial words rarely or frequently appear in proxy logs, might not contribute the detection rate. To generate discriminative feature representation, a corpus has to be effectively summarized. To summarize the corpus effectively, the important words for classification have to be extracted.

Our method extracts important words from proxy logs to summarize the corpus. This technique is known as undersampling or bandpass sampling. To define the word importance score, our method uses term frequency and document frequency. Our method summarizes the corpus and improves the detection rate. This paper demonstrates that our method improves the accuracy with actual proxy logs. Our method extracts important words from proxy logs, which are based on 3 indicators: TF (Term Frequency), DF (Document Frequency) and TFIDF (Term Frequency Inverse Document Frequency). TFIDF is a numerical statistic that is intended to reflect word importance. Our method can detect unseen malicious traffic in actual proxy logs.

Furthermore, we perform experiments with actual proxy logs. In the previous work, we mingled malicious and benign pcap files, and verified that the previous method could detect unseen malicious traffic. Malicious

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

traffic, however, accounts for only a small portion in actual proxy logs. Therefore, the previous method has to be evaluated in more practical conditions with actual proxy logs. In this paper, we realistically mingle these logs into datasets without adjusting the size.

The main contributions of this paper are as follows:

(1) Propose three methods to improve a linguistic-based detection method in proxy logs[2].
(2) Demonstrate that the previous method is not effective in actual proxy logs because of the imbalance[3].
(3) Show that our method can detect unseen malicious traffic in actual proxy logs.

The rest of the paper is organized as follows. Next section describes Natural Language Processing (NLP) techniques. Section 3 introduces the linguistic-based detection method and indicates the problem. Section 4 proposes a method to improve the linguistic-based detection method. Section 5 demonstrates that dominant benign traffic occupies malicious feature representation, and shows the effectiveness of our method. Section 6 evaluates the result and section 7 discusses related work.

2. Natural Language Processing (NLP) Technique

2.1 Paragraph Vector (Doc2vec)

Paragraph Vector (Doc2vec)[4] is an extension of Word2vec, and constructs embedding from entire documents. Word2vec is a shallow two-layer neural network that is trained to reconstruct linguistic contexts of words. Word2vec has two models to produce a distributed representation of words. Continuous-Bag-of-Words (CBoW) model predicts the current word from a window of surrounding context words. Skip-gram model uses the current word to predict the surrounding window of context words with the order reversed. These models enable to calculate the semantic similarity between two words and infer similar words semantically. The same idea has been extended to entire documents. In the same manner, Doc2vec has two models to produce Paragraph Vector a distributed representation of entire documents. Distributed-Memory (DM) is the extension of CBoW, and the only change is adding a document ID as a window of surrounding context words. Distributed-Bag-of-Words (DBoW) is the extension of skip-gram, and the current word was replaced by the current document ID. Doc2vec enables to calculate semantic similarity between two documents and infer similar documents semantically. Doc2vec also supports inference of document embedding on unseen documents. The linguistic-based detection method[1] uses this function to represent unseen traffic.

2.2 TFIDF (Term Frequency Inverse Document Frequency)

TFIDF is a numerical statistic that is intended to reflect word importance to a document in a collection or corpus, and one of the most popular term-weighting schemes. A TFIDF score increases proportionally to the number of times a word appears in the document, and is often offset by the frequency of the word in the corpus. TFIDF is the product of two statistics, term frequency and inverse document frequency. Term Frequency (TF) is the number of times that a term occurs in a document. Inverse Document Frequency (IDF) is a measure of how much information the term provides. This means whether the term is common or rare across all documents.

TFIDF of term i in document j in a corpus of D documents is calculated as follows.

$$TFIDF_{i,j} = frequency_{i,j} \times \log_2 \frac{D}{document\_frequency_i}$$

A high weight in TFIDF is reached by a high term frequency in the given document and a low document frequency of the term in the whole collection of documents. Therefore, the weights tend to exclude common terms. The ratio inside the log function is always greater than or equal to 1. Hence, the TFIDF score is greater than or equal to 0. As a term appears in more documents, the ratio inside the logarithm approaches 1, bringing the TFIDF score closer to 0.

Both high term frequency and low document frequency are important for classification. Therefore, our method utilizes TFIDF scores to extract important words from proxy logs.

3. Linguistic-based Detection Method

3.1 Extracting words

Based on a linguistic approach, a malicious traffic detection method for proxy logs has been proposed[1]. The key idea of this method is treating proxy logs as a natural language. **Fig. 1** shows a sample of proxy logs.

This sample contains date and time at which transaction completed, request line from the client (includes the method, the URL and the user agent), HTTP status code returned to the client and size of the object returned to the client. The client is user's computer which connects to servers over the Internet. A proxy server records the

```
[29/Sep/2016:15:03:06 +0000] "" 10.9.29.102 200 "GET http://edu.xxx.com/index.php
HTTP/1.1" "" "" "" 979 "Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like
Gecko" "" ""
[29/Sep/2016:15:05:43 +0000] "" 10.9.29.102 502 "GET
http://pmenboeqhyrpvomq.yyy.bid/0123-4567-89AB-CDEF-0123?auto HTTP/1.1" "" "" "" 341
"Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko" "" ""
[29/Sep/2016:15:44:07 +0000] "" 10.9.29.103 200 "GET http://en.zzz.us/index.php HTTP/1.1"
"" "" "" 995 "Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko" "" ""
[29/Sep/2016:16:15:57 +0000] "" 10.9.29.104 200 "GET http://en.zzz.us/index.php HTTP/1.1"
"" "" "" 1000 "Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko" "" ""
```

Fig. 1 A sample of proxy logs.

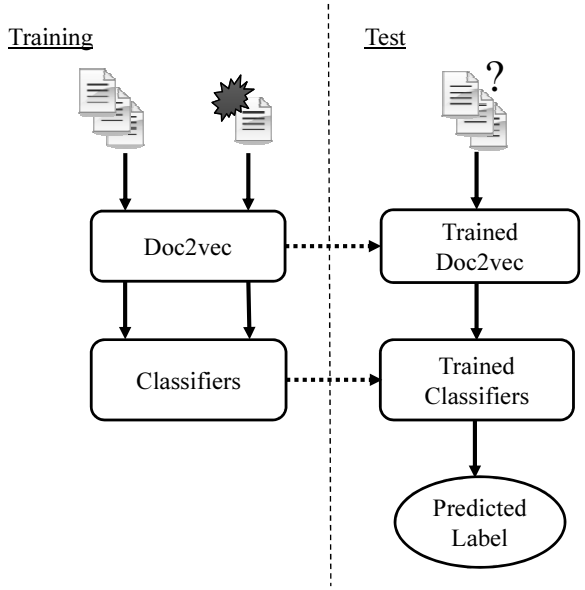

Fig. 2 The linguistic-based detection method (Previous Method).

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.

This method divides a single log line into HTTP status code, request line from the client, size of the object returned to the client and user agent by a single space. Then, the request line is divided into method, URL and protocol version by a single space. Furthermore, this method divides the URL into words by the delimiters which are "dot" (.), "slash" (/), "question mark" (?), "equal" (=) and "and" (&). This method treats each strings separated by a single space or the delimiters as a word. A single paragraph consists of the words extracted from consecutive 10 log lines. The consecutive log lines do not include overlapping. The appropriate number of log lines was empirically determined. Thus, this method derives words and paragraphs from proxy logs.

3.2 Overview

Fig. 2 shows an overview of the previous method[1].

This method uses two types of machine learning techniques. One is Doc2vec an unsupervised learning model for feature extraction. This method constructs a Doc2vec model from the paragraphs and trains the model. The paragraphs are derived from benign and malicious proxy logs. Note that these logs have to be known as benign or malicious. Then, the paragraphs are converted into feature vectors. In this process, the feature representation is automatically extracted by the trained Doc2vec model, that is why this method utilizes the neural network model. The other is a supervised learning model to classify the feature vectors. This method trains the classifiers with the feature vectors and labels.

Subsequently, this method derives paragraphs from unknown proxy logs. The paragraphs are converted into feature vectors with the trained Doc2vec model. Note that the Doc2vec model is not trained by the paragraphs extracted from unknown proxy logs. Finally, the trained classifiers predict the labels from the feature vectors.

3.3 Imbalance

The previous method[1] utilizes Paragraph Vector to convert words into feature vectors automatically. In the previous method, the converted feature vectors are key factors to discriminate between benign traffic and malicious traffic. To generate discriminative feature vectors, an appropriate corpus is required. The previous method, however, generates a corpus from the whole extracted words from proxy logs. Certainly, the previous method performed good accuracy in the datasets, which were generated from benign and malicious pcap files. The previous method, however, requires affordable benign and malicious corpuses continually. Because new websites are being created constantly as with malware. Then, the benign corpus is extracted from proxy logs in each organization. The malicious corpus is extracted from malicious pcap files which are downloaded from the websites such as Malware-Traffic-Analysis.net[*1]. In reality, proxy logs in a large organization contain a huge amount of words. In contrast, these malicious pcap files contain only limited words. If we generate a dataset from both words at the same ratio, the generated corpus cannot represent benign feature adequately. Because the corpus contains only benign words as much as words in malicious corpus. If we merely generate a dataset from both whole words, an imbalance occurs. In this case, benign words occupy most of the corpus. Such a corpus cannot generate discriminative feature representation. Thus, benign traffic is dominant, and occupies malicious feature representation. Therefore,

[*1] http://www.malware-traffic-analysis.net/

this method might not perform accuracy in practical environment. A further section describes the detail. To perform accuracy in practical environment, this method requires a balanced corpus.

4. Proposed Method

4.1 Key Idea

To mitigate the imbalance, we pursue how to generate a balanced corpus from actual proxy logs. In general, benign proxy logs contain a huge amount of words. In contrast, malicious pcap files contain only limited words. To generate a balanced corpus, both numbers of unique words have to be adjusted at the same ratio. The previous method extracts whole words from proxy logs without selecting important words. Hence, selecting important words might mitigate the imbalance.

To select important words, we focus on Term frequency (TF), Document frequency (DF) and TFIDF. In our method, TF is the number of times that a word appears in whole proxy logs. DF is the number of paragraphs where a word appears. To calculate these scores, we interpret whole malicious logs and benign logs as each document respectively. Because our purpose is discriminating between malicious and benign logs. Our purpose is neither identifying each paragraph nor classifying paragraphs. Our method selects important unique words based on the importance scores.

Furthermore, the important unique words are selected from malicious and benign logs respectively. To discriminate between malicious and benign logs, the unique words in each logs have to be selected respectively. If the important unique words are selected from only benign logs, it is difficult to discriminate between malicious and benign logs. Therefore, our method selects the important unique words from benign and malicious logs equally. To perform this task, both numbers of unique words have to be adjusted at the same rate. To adjust the number of unique words, we utilize the word importance scores. Our method extracts important words based on the word importance scores, and removes the trivial words.

In this manner, we extract important words from proxy logs to summarize the corpus. Our method summarizes the corpus and mitigates the imbalance.

4.2 Overview

In this paper, we propose a modified linguistic-based detection method which includes how to extract important words and how to generate an effective corpus. **Fig. 3** shows an overview of the proposed method.

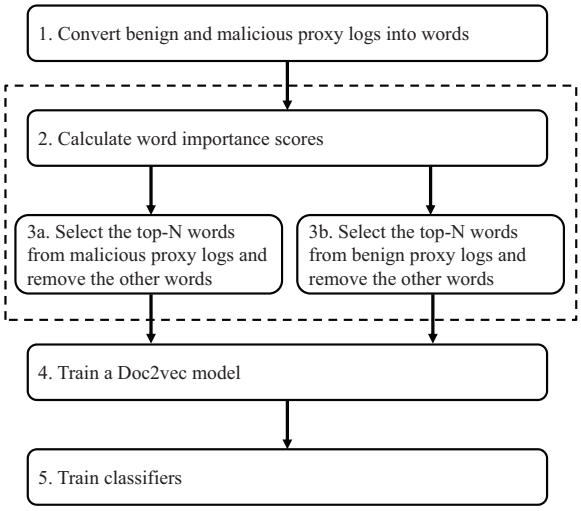

Fig. 3 An overview of the proposed method.

Updated processes are surrounded by a broken line. First, our method converts malicious and benign proxy logs or pcap files into words (1). Pcap files are converted into proxy logs. Both logs are separated by a single space or the delimiters. A single paragraph consists of the words extracted from 10 log lines. We adopted the same number of log lines for comparison with the previous method. Next, our method calculates word importance scores (TF, DF, TFIDF) from the malicious and benign words (2), and extracts top-N important words respectively (3ab). The extracted top-N unique words are different in each traffic. These unique words expect to equally represent both traffic. Note that both numbers of the extracted words are equal. To construct an effective corpus, both numbers of unique words have to be the same number. That is why our method extracts top-N important words, does not extract words above a threshold. The default value for N is the half number of unique words which are extracted from the malicious proxy logs. Our method extracts important words based on the word importance score, and removes trivial words. Thus, our method mingles both words, and obtains an effective corpus. Then, our method trains a Doc2vec model with the effective corpus, and converts both words into feature vectors with the trained model (4). Finally, our method trains classifiers with the feature vectors and labels (5). 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 in 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 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.

Subsequently, our method derives paragraphs from unknown proxy logs in the same way. This method extracts the top-N important words from the paragraphs, which were extracted in the training process. The modified paragraphs are converted into feature vectors with the trained Doc2vec model. Then, the trained classifiers predicts the label from the feature vectors. The predicted label is either malicious or benign.

4.3 Implementation

We implemented our method with gensim-1.2.1[*2], scikit-learn-0.19.1[*3] and chainer-2.0.12[*4].

Gensim is a Python library to realize unsupervised semantic modelling from plain text, and provides a Doc2vec model. We set the same parameters for comparison with the previous method[1]. Scikit-learn is a machine-learning library for Python that provides tools for data mining with a focus on machine learning. We use a SVC function with a linear kernel for SVM and a RandomForestClassifier function for RF. Chainer is a flexible Python framework for neural networks. Our method trains a classifier 30 epochs. The classifier defines the loss function in optimization with ReLU (Rectified Linear Unit), Adam (Adaptive moment estimation) and cross entropy.

5. Experiment

5.1 Dataset and environment

In this experiment, we use captured pcap files from Exploit Kit (EK) between 2014 and 2017 as malicious traffic. EK is a software kit designed to run on web servers, with the purpose of identifying vulnerabilities in client computers. EK seeks and exploits vulnerabilities to upload and execute malicious code on the client computer. We chose some EKs which communicate via a standard protocol and attempt to imitate benign http communication, and downloaded the packet traces from the website MALWARE-TRAFFIC-ANALYSIS.NET. **Tab. 1** shows the detail.

Tab. 1 The detail of the MTA dataset.

period	size	number	description
2014	238M	258	Angler, Fiesta, FlashPack, Magnitude Neutrino, Nuclear, RIG
2015	186M	161	Angler, Fiesta, Magnitude, Neutrino Nuclear, RIG
2016	373M	406	Angler, Magnitude, Neutrino, Nuclear RIG
2017	109M	69	RIG

Tab. 2 Experimental environment

CPU	Core i9-7900X 3.3GHz
Memory	DDR4 2666 128GB
GPU	GeForce GTX1080/11G
OS	Windows 10

This dataset (MTA) contains the pcap files extracted from traffic of the EKs. Our method aims to detect malicious traffic from proxy logs. Thus, we converted these pcap files into proxy logs. We extracted http traffic from these pcap files and concatenated the requests and responses.

In this paper, we use actual proxy logs between 2016 and 2017 as benign traffic. These actual proxy logs were collected at a campus network, which belongs to a class B network. This network consists of more than 5,000 computers. We extracted 1G bytes of log in April 2016 as the training data, and 9.65G bytes of log in April 2017 as the test data. Several experts confirmed that these proxy logs were benign with many security devices. We assume that these proxy logs represent benign traffic.

We mingle the malicious logs and benign logs into datasets. We split the datasets into training data and test data for timeline analysis. This paper uses three metrics: Precision (P), Recall (R) and F-measure (F). The experimental environment is shown in **Tab. 2**.

[*2] https://radimrehurek.com/gensim/
[*3] http://scikit-learn.org/
[*4] https://chainer.org/

Tab. 3 The result of the 10-fold cross-validation.

method	classifier	Benign			Malicious		
		P	R	F	P	R	F
Previous Method	SVM	0.99	0.99	0.99	0.53	0.90	0.65
	RF	0.99	0.99	0.99	0	0	0
	MLP	0.99	0.99	0.99	0.80	0.89	0.83
Proposed Method (TF)	SVM	0.99	0.99	0.99	0.92	0.92	0.91
	RF	0.99	1.00	0.99	1.00	0.67	0.79
	MLP	0.99	0.99	0.99	0.99	0.92	0.95
Proposed Method (DF)	SVM	0.99	0.99	0.99	0.94	0.92	0.93
	RF	0.99	1.00	0.99	1.00	0.94	0.77
	MLP	0.99	0.99	0.99	0.99	0.93	**0.96**
Proposed Method (TFIDF)	SVM	0.99	0.99	0.99	0.93	0.94	0.94
	RF	0.99	1.00	0.99	1.00	0.14	0.23
	MLP	0.99	0.99	0.99	0.99	0.93	**0.96**

Tab. 4 The numbers of unique words and vectors.

method	class	dataset		
		train		test
		word (N)	vector	
Previous Method	Malicious	7,629	233	367
	Benign	608,796	32,158	3,357,841
Proposed Method	Malicious	3,815	233	367
	Benign	3,815	32,158	3,357,841

5.2 Cross-validation

To compare the previous method[1] with the proposed method in basic performance, we performed 10-fold cross-validation with 2014's MTA and the 1G bytes of logs in April 2016. **Tab. 3** shows the result.

All methods perform good accuracy in the benign traffic. In the malicious traffic, the three proposed methods yield better performance than the previous method. The best F-measure has reached 0.96.

5.3 Timeline Analysis

Next, we performed timeline analysis. In the timeline analysis, we used 2015's MTA and the 1G bytes of logs in 2016 as the training data, and 2016's MTA and the 9.65G bytes of logs in April 2017 as the test data. First, **Tab. 4** shows the numbers of unique words and vectors.

The previous method does not extract important words. The numbers of original unique words in malicious and benign traffic are 7,629 and 608,796 respectively. The previous method constructed a Doc2vec model from the original words. On the other hand, the proposed method extracted 3,815 important words from malicious and benign traffic respectively, and constructed a Doc2vec model from the reduced words. The numbers of vectors by both methods are the same, and the numbers of the malicious vectors and benign vectors are quite different. This result shows that our datasets are extremely imbalanced. This means the experimental environment is more fair and practical.

Tab. 5 Performance of the timeline analysis.

method	classifier	Benign			Malicious		
		P	R	F	P	R	F
Previous Method	SVM	0.99	0.98	0.99	0	0.74	0.01
	RF	0.99	0.99	0.99	0	0	0
	MLP	0.99	0.99	0.99	0.01	0.61	0.02
Proposed Method (TF)	SVM	0.99	0.99	0.99	0.19	0.90	0.31
	RF	0.99	0.99	0.99	0.98	0.51	0.67
	MLP	0.99	0.99	0.99	0.97	0.93	**0.95**
Proposed Method (DF)	SVM	0.99	0.99	0.99	0.16	0.90	0.27
	RF	0.99	0.99	0.99	0.99	0.54	0.70
	MLP	0.99	0.99	0.99	0.89	0.91	0.90
Proposed Method (TFIDF)	SVM	0.99	0.99	0.99	0.67	0.93	0.78
	RF	0.99	0.99	0.99	0.96	0.19	0.32
	MLP	0.99	0.99	0.99	0.67	0.78	0.72

Next, **Tab. 5** shows performance of the timeline analysis.

In spite of reducing words, the performance of the benign traffic was almost satisfactory. We focus on the malicious traffic. The three proposed methods maintained the performance to a degree in the timeline analysis. The performances of the proposed method were better than the previous method. The best F-measure has reached 0.95. MLP achieved good accuracy on average. RF were less accurate than the other classifiers. In contrast, the previous method shows quite poor performance. This is because dominant benign traffic occupies malicious feature representation. Recall that the number of unique words in benign traffic was too larger than the number in malicious traffic. Thus, the previous method does not perform accuracy in practical environment. The proposed method could detect unseen malicious traffic in actual proxy logs.

Tab. 6 shows the required time of the timeline analysis.

In training phase, the previous method requires too much time. This is because the previous method uses the whole words (616,425) to construct a Doc2vec model, whereas the proposed methods uses only important words

Tab. 6 Required time of the timeline analysis.

method	classifier	training	test
Previous	SVM	1h 34m 54s	7m 43s
Method	RF	1h 14m 44s	8s
	MLP	4h 10m 7s	3m 34s
Proposed	SVM	32m	4m 47s
Method	RF	23m 8s	6s
(TF)	MLP	3h 8m 19s	3m 20s
Proposed	SVM	31m 37s	4m 45s
Method	RF	22m 31s	16s
(DF)	MLP	3h 29m 29s	3m 55s
Proposed	SVM	53m 50s	7m 31s
Method	RF	28m 11s	7s
(TFIDF)	MLP	3h 52m 6s	3m 20s

Tab. 7 Performance of the long-term analysis.

method	training data	test data	Malicious		
			P	R	F
TF	2014	2015	0.97	0.93	0.95
DF			0.89	0.91	0.90
TFIDF			0.67	0.78	0.72
TF	2014	2016	0.80	0.92	0.86
DF			0.93	0.93	0.93
TFIDF			0.83	0.79	0.81
TF	2014	2017	0.85	0.71	0.78
DF			0.71	0.69	0.70
TFIDF			0.28	0.53	0.36

(7,630). MLP requires several hours for training. SVM and RF require several tens of minutes. In test phase, SVM and MLP require 3 to 7 minutes to analyze logs whose sizes are more than 9.65G bytes. RF require only a few seconds to analyze the logs.

Furthermore, we performed the long-term analysis. According to the result of the timeline analysis, MLP is the best classifier. Therefore, our method uses MLP as the classifier. **Tab. 7** shows performance of the long-term analysis.

Contrary to our expectation, TF and DF were better than TFIDF. This result implies a combination of common words is more effective rather than rare words. Therefore, we concluded simple methods were effective to extract important words from proxy logs.

6. Discussion

6.1 Accuracy

In this experiment, we performed cross-validation and timeline analysis. We realistically mingled the malicious and benign logs into datasets without adjusting the size. The benign logs had the majority in the datasets. Hence, we believe this condition is practical. According to the result, the best classifier was MLP and the best F-measure has reached 0.95. Therefore, we conclude our methods can detect unseen malicious traffic in the practical condition. Our methods are superior to the previous method[1], especially in practical conditions. This is because our methods extracted important words from malicious and benign traffic respectively, and constructed a Doc2vec model from the essential words.

6.2 Required Time

In the timeline analysis, our method could construct trained models in less time than the previous method. Our method requires several hours or several tens of minutes with 1G bytes of log for training. In practical use, we can train these models before real time network monitoring.

Then, our method could complete analysis of massive proxy logs in a small amount of time. Our method requires only several minutes to analyze logs whose sizes are more than 9.65G bytes. Therefore, our method performs enough fast for real time network monitoring.

6.3 Ethics

In this paper, we used malicious pcap files and benign proxy logs. These files might contain privacy sensitive information such as personal information, email addresses or 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 or email addresses. Our method does not require monitoring all network traffic. Furthermore, our method does not require client's IP addresses and even distinguishing the client's sources. This means our method accepts most vantage points to monitor traffic.

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

sistance.

7. Related Work

Our method uses only proxy logs, and does not require any additional information obtained from the outside.

Invernizzi et al.[7] built a network graph with IP addresses, domain names, FQDNs, URLs, paths and file names. Their method focuses on the correlation among nodes to detect malware distribution. In this method, the whole parameters are obtained from proxy logs. This method, however, has to cover many ranges of IP addresses, and performs in large-scale networks such as ISPs. In addition, this method requires the downloaded file types. Nelms et al.[8] 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. Bartos et al.[9] categorized proxy logs into flows, and extracted various features from the flows. They proposed how to learn the feature vectors to classify malicious URLs. This method can decide the optimum feature vectors automatically. However, this method demands devising basic features for learning. Mimura et al.**?** categorized proxy 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. This method, however, performs for only C&C traffic. Shibahara et al.[11] focus on a sequence of URLs which include malicious artifacts of malicious redirections, and built a detection system which uses Convolutional Neural Networks. This method uses a honey client to collect URL sequences and their labels, and performs for DbD attacks. Our method uses only proxy logs and does not require any additional information and does not require devising feature vectors at all. In addition, our method performs at any scale and can detect not only DbD attacks but also C&C traffic.

8. Conclusion

In this paper, we focus on that the previous method[1] generates a corpus from the whole extracted words which contain trivial words. This paper demonstrates that the previous method is not effective in actual proxy logs because of the imbalance. To mitigate the imbalance, we propose how to generate a balanced corpus from actual proxy logs. Our method extracts important words from proxy logs. The experimental results show our method could detect unseen malicious traffic in the practical environment. The best F-measure achieves 0.95 in the timeline analysis.

In this paper, we used the datasets which were generated by mixing malicious pcap files and benign proxy logs. Applying our method to other proxy logs is a future work. Another future work is how to update the model. New training data did not always improve the accuracy. As previously mentioned before, benign and malicious corpuses continually have to be updated.

参考文献

[1] Mimura, M., and Tanaka, H., *Leaving All Proxy Server Logs to Paragraph Vector* Journal of Information Processing, Vol.26, pp.804-812 (2018).

[2] Mimura, M., *On the Effectiveness of Extracting Important Words from Proxy Logs* Proc. 5th International Workshop on Information and Communication Security, pp.424-430 (2018).

[3] Mimura, M., and Tanaka, H., *A Linguistic Approach towards Intrusion Detection in Actual Proxy Logs* Proc. 20th International Conference on Information and Communications Security, LNCS, Vol.11149, pp.708-718 (2018).

[4] Le, Q., and Mikolov, T., *Distributed Representations of Sentences and Documents*, Proc. 31st International Conference on Machine Learning, pp.1188-1196 (2014).

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

[6] 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).

[7] 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).

[8] 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).

[9] 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).

[10] 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).

[11] Shibahara, T., Yamanishi, K., Takata, Y., Chiba, D., Akiyama, M., Yagi, T. Ohsita, Y., and Murata, M., *Malicious URL Sequence Detection using Event Denoising Convolutional Neural Network* Proc. 2017 IEEE International Conference on Communications, pp.1-7 (2017).