# Reading Network Packets as a Natural Language for Intrusion Detection

Mamoru Mimura(✉) and Hidema Tanaka

National Defense Academy, 1-10-20 Hashirimizu, Yokosuka, Kanagawa, Japan
mim@nda.ac.jp

**Abstract.** Detecting unknown malicious traffic is a challenging task. There are many behavior-based detection methods which use the characteristic of drive-by-download attacks or C&C traffic. However, many previous methods specialize the attack techniques. Thus, the adaptability is restricted. Moreover, they need to decide the feature vectors every attack method. This paper proposes a generic detection method which does not depend on attack methods and does not need devising feature vectors. This method reads network packets as a natural language with Paragraph Vector an unsupervised algorithm, and learns the feature automatically to detect malicious traffic. This paper conducts timeline analysis and cross-dataset validation with the multiple datasets which contain captured traffic from Exploit Kit (EK). The best F-measure achieves 0.98 in the timeline analysis and 0.97 on the other dataset. Finally, the result shows that using Paragraph Vector is effective on unseen traffic in a linguistic approach.

**Keywords:** Drive by download · C&C · Neural network
Bag of Words · Word2vec · Paragraph Vector · Doc2vec
Support Vector Machine

## 1 Introduction

In recent years, Drive-by Download attack (DbD attack) and Spear Phishing attack are main attack techniques on the Internet. In general, intrusion detection techniques on a network are classified roughly into pattern-matching-based methods and methods using blacklists. The pattern-matching-based methods are effective, if the malicious traffic contains a unique string pattern. IDS uses fixed strings or regular expression to describe the signatures. However, recent Exploit Kits (EKs) (e.g. Angler EK, RIG EK) communicate via a standard protocol to imitate benign http traffic. 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 to upload and execute malicious code on the client via standard protocols. Some queries do not contain the EK specific strings. Therefore, it is difficult to describe the signatures. In this case, IDS can use the malicious destination server (e.g. Landing 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, the attacker can change the malicious destination servers to evade detection by network devices. Some attackers use compromised hosts as stepping stones. Therefore, using the blacklist does not play a critical role. In addition, the malicious server address has to be already-known before the cyberattack. Thus, detecting unknown malicious traffic is a challenging task.

There are many behavior-based detection methods to detect unknown malicious traffic. These methods capture the characteristics of DbD attacks [22–24] or C&C traffic [9,10], and detect unseen malicious traffic. Many previous methods, however, can detect only DbD attacks or C&C traffic. Because these methods usually use different detection techniques. If attackers modify the attack techniques, these previous methods barely detect unseen malicious traffic. Besides security researchers need to devise the feature vectors to capture the characteristics. Furthermore, some attackers still use protocols other than http or https. For instance, recent WannaCry ransomware uses Server Message Block (SMB) to compromise Windows machines, load malware, and propagate to other machines in a network. SMB is a transport protocol used by Windows computers for a wide variety of purposes such as file or printer sharing.

This paper focuses on the characteristic that Neural Network (NN) learns feature vector representation automatically. In this paper, we presume network packets are written in a natural language, and attempt to learn the difference of benign traffic and malicious traffic automatically with Paragraph Vector. Paragraph Vector is an unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of texts. Then we input the extracted feature vectors with the label into supervised learning models to classify benign traffic and malicious traffic. The key idea of this research is reading network packets as a natural language.

This paper proposes a generic detection method, which does not depend on attack methods and does not need devising feature vectors. Our generic detection method does not rely on protocols and attack techniques, and does not demand devising feature vectors. This paper conducts timeline analysis with the dataset which contains captured traffic from Exploit Kit (EK) between 2014 and 2017. We used the most up-to-date captured traffic which was downloaded from the website Malware-Traffic-Analysis.net [1]. This paper also demonstrates cross-dataset validation by showing that an automated feature extraction scheme learned from one dataset can be used successfully for classification on another dataset. The best F-measure achieves 0.98 in the timeline analysis and 0.97 on the other dataset. Finally, the result shows that using Paragraph Vector is effective on unseen traffic in a linguistic approach.

The main contributions of this paper are four-fold: (1) Proposed a generic detection method which did not rely on protocols and attack techniques. (2) Verified that the proposed method could detect up-to-date EKs. (3) Verified that the proposed method was effective on another dataset. (4) Verified that using Paragraph Vector is effective on unseen traffic in a linguistic approach.

The rest of the paper is organized as follows. Next section briefly discusses related works and makes clear the difference among our method and previous methods. Section 3 describes Natural Language Processing (NLP) techniques which include Paragraph Vector. Section 4 proposes a generic detection method based on the NLP techniques. Section 5 shows experimental results applying the proposed method to the multiple datasets. Section 6 discusses the results, and reveals the performance and effectiveness.

## 2 Related Work

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 barely detects unknown malicious traffic. Therefore, many behavior-based detection methods are proposed. For example, some methods focused on the traffic classification from packet traces [4–8]. Analyzing packets is, however, 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 [9,10], DNS records [11–14] and proxy server logs [15–22]. However, recent Exploit Kits (EKs) (e.g. Angler EK, RIG EK) communicate via a standard protocol to imitate normal http communication. Furthermore, some attackers use compromised hosts as stepping stones. Thus, detecting recent EKs from logs is becoming a challenging task. Therefore, many methods use additional contents to distinguish malicious traffic from seemingly benign traffic. For instance, some methods analyze JavaScript code to detect these EKs [23,24].

These previous methods utilize the characteristic of DbD attacks or C&C traffic well. However, their good contrivance can backfire. Many previous methods specialize the attack techniques, and the adaptability is limited. In addition, many previous methods using machine learning technique demand devising feature vectors to distinguish malicious traffic. In other words, the essence of the previous works was how to extract feature vectors from network traffic, logs and contents. Our method is fundamentally different and based on the other viewpoint. Our method is a generic intrusion detection method which can detect many attack techniques with a simple technique. Our method does not demand devising feature vectors. Because our method learns the difference of benign traffic and malicious traffic automatically with NN.

## 3 Natural Language Processing (NLP) Technique

### 3.1 Word2vec

To calculate various measures to characterize a text, we have to transform the text into a vector. Word2vec [2] 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 in semantics or syntactic. 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 enables to calculate the semantic similarity between two words and infer similar words semantically. However, Word2vec is a model that merely produces word embedding. To calculate the semantic similarity between two documents, this model has to be extended.

### 3.2 Paragraph Vector (Doc2vec)

An extension of Word2vec to construct embedding from entire documents has been proposed [3]. 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 enables to calculate the 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 to develop 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 Translating Network Packets into a Language

The key idea of our method is reading network packets as a natural language. In order to apply NLP techniques, network packets have to be translated and

```
192.168.204.175 → 206.188.192.114 TCP 66 49380 → 80 [SYN] Seq=0 Win=8192
Len=0 MSS=1460 WS=4 SACK_PERM=1
206.188.192.114 → 192.168.204.175 TCP 60 80 → 49380 [SYN, ACK] Seq=0 Ack=1
Win=64240 Len=0 MSS=1460
192.168.204.175 → 206.188.192.114 TCP 60 49380 → 80 [ACK] Seq=1 Ack=1
Win=64240 Len=0
```

**Fig. 1.** Summary lines in a transport layer.

```
10.180.0.14 → 10.180.0.254 DNS 82 Standard query 0xe854 A www.antiqueceramics.tk
10.180.0.14 → 10.180.0.254 DNS 82 Standard query 0xe854 A www.antiqueceramics.tk
10.180.0.254 → 10.180.0.14  DNS 371 Standard query response 0xe854 A
www.antiqueceramics.tk CNAME antiqueceramics.tk A 195.20.34.1 A 195.20.34.2 NS
d.ns.tk NS c.ns.tk NS b.ns.tk NS a.ns.tk A 194.0.41.1 AAAA 2001:678:5c::1 A 194.0.39.1
AAAA 2001:678:54::1 A 194.0.40.1 AAAA 2001:678:58::1 A 194.0.38.1 AAAA
2001:678:50::1
10.180.0.14 → 195.20.34.1  HTTP 316 GET / HTTP/1.0
195.20.34.1 → 10.180.0.14  HTTP 60 HTTP/1.0 203 Non-Authoritative Information
(text/html)
```

**Fig. 2.** Summary lines in an application layer.

separated into words. However, reading network packets is becoming intractable on broadband networks. Therefore, we need a lightweight translation method. To translate network packets into a language, we use TShark [25] a network protocol analyzer, which captures and decodes packet data from a network. TShark displays a summary line for each received packet. The summary line consists of some fields such as source and destination IP address, protocol, size and basic contents. Our method uses these fields as separated words.

Figure 1 shows summary lines in a transport layer. In a transport layer, our method uses protocol, port number, size and flags. Our method ignores other parameters, because there are a very wide range of the types.

Figure 2 shows summary lines in an application layer. In an application layer, our method uses all fields after protocol. Furthermore, our method separates FQDN (Fully Qualified Domain Name) by "dot" (.). Then we can derive top level domain name, sub domain name and so on, which means the country, organization, use or the purpose (e.g. www, mail). Our method separates path by "slash" (/) and "dot" (.), "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.

### 4.2 Proposed Method

Figure 3 shows an overview of our method. First, our method constructs a corpus from known malicious traffic and benign traffic. Each traffic is translated and separated by the previously mentioned method. In this paper, we use 2 reading methods. Table 1 shows a summary of the reading methods. Each method reads only each layer, and collects 100 summary lines for a paragraph.

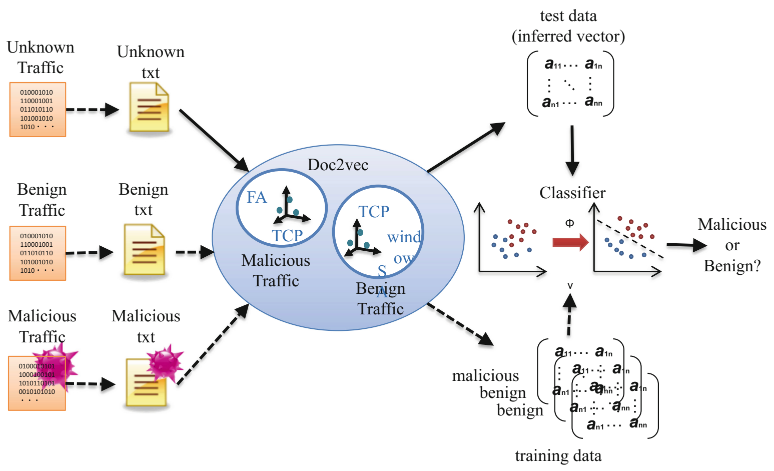

**Fig. 3.** An overview of the proposed method.

**Table 1.** A summary of the reading methods

| Method | Layer | Contents |
|---|---|---|
| Method 1 | Transport layer | Protocol, port number, size |
| Method 2 | Application layer | FQDN, path, user agent |

Then, the Doc2vec constructs a vector space from the corpus, and converts each paragraph into vectors with the labels. These labeled vectors are training data for a classifier. In this time, we use Support Vector Machine (SVM) for the classifier. 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 mapped into that same space and predicted to belong to a category based on which side of the gap they fall. Given a set of training data, each labeled as belonging to one or the other of two categories, this training algorithm builds a model that assigns new examples to one category or the other.

After that, we convert unknown traffic into vectors. These unlabeled vectors are test data for the classifier. Finally, we input these unlabeled vectors to the trained classifier, and can obtain a predicted label. The predicted label is either malicious or benign.

## 4.3 Implementation

The proposed method was developed by Python-2.7 with open source machine learning libraries, gensim-1.01 [26] and scikit-learn-0.18.0 [27].

Gensim is a Python library to realize unsupervised semantic modelling from plain text, and includes a Doc2vec model. Table 2 shows the parameters for the

**Table 2.** The parameters for the Doc2vec model

| | |
|---|---|
| Dimensionality of the feature vectors | 100 |
| Window | 15 |
| Number of epochs | 30 |
| Training algorithm | DBoW |

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. Scikit-learn is a machine-learning library for Python that provides tools for data mining with a focus on machine learning. The proposed method uses a SVC function with a linear kernel for SVM.

## 5 Experiment

### 5.1 Dataset

To reveal the effectiveness to up-to-date EKs, we use captured pcap files from EKs between 2014 and 2017 (MTA dataset), which were downloaded from the website Malware-Traffic-Analysis.net [1]. We also use D3M (Drive-by Download Data by Marionette) dataset and NCD (Normal Communication Data in MWS-Cup 2014) for the cross-dataset validation. These datasets are parts of MWS datasets [28], and include pcap files. MTA and D3M contain malicious traffic and NCD contains benign traffic. Table 3 shows the detail.

The MTA dataset contains traffic from the following EKs, Angler EK, Fiesta-EK, FlashPack-EK, Magnitude EK, Neutrino EK, Nuclear EK and RIG EK. 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. This dataset contains traffic from EKs (e.g. Blackhole EK, Elenore, Mpack).

**Table 3.** The detail of the datasets.

| MTA | | D3M | |
|---|---|---|---|
| Year | Size | Year | Size |
| 2014 | 238M | 2010 | 130M |
| 2015 | 186M | 2011 | 24.8M |
| 2016 | 373M | 2012 | 33.2M |
| 2017 | 109M | 2013 | 14.6M |
| - | - | 2014 | 23.3M |
| - | - | 2015 | 334M |

We extracted summary lines from these pcap files with TShark. After that, we compounded the malicious summary lines and the benign summary lines into a dataset at the same rate. We split the dataset into training data and test data to conduct timeline analysis and cross-dataset validation. The proposed method uses only training data to construct a corpus. Because, our method presumes that the test data is completely unknown traffic in practical condition. To compare the characteristics of our method, we also use a Bag-of-Words (BoW) model. BoW is a simplifying representation used in natural language processing. The most common type of features calculated from BoW is the number of times a term appears in the sentence. In the timeline analysis, we chose an annual traffic as training data, and the subsequent traffic is the test data.

## 5.2 Result

In this experiment, we use 3 metrics: Precision (P), Recall (R) and F-measure (F). Table 4 shows the results of the timeline analysis.

Contrary to expectations, BoW was generally more effective than Doc2vec with the method 1. The both F-measures maintain a generally constant values over three years, and the best one has reached 0.96. This result means that word frequency is the most distinctive feature in a transport layer.

Doc2vec was generally more effective than BoW with the method 2. The best F-measure has reached 0.98 in the next year, and the both F-measures gradually decrease. This result means that Doc2vec capture distinctive features other than word frequency in an application layer.

**Table 4.** The result of the timeline analysis.

| Method | Training data | Test data | Model | NCD (Benign) | | | MTA (Malicious) | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | P | R | F | P | R | F |
| 1 | 2014 | 2015 | BoW | 0.97 | 0.93 | 0.95 | 0.93 | 0.97 | 0.95 |
| | | | Doc2vec | 0.94 | 0.80 | 0.86 | 0.93 | 0.95 | 0.88 |
| | | 2016 | BoW | 1.00 | 0.92 | 0.96 | 0.92 | 1.00 | 0.96 |
| | | | Doc2vec | 0.97 | 0.78 | 0.87 | 0.82 | 0.97 | 0.89 |
| | | 2017 | BoW | 0.99 | 0.92 | 0.95 | 0.92 | 0.99 | 0.96 |
| | | | Doc2vec | 0.99 | 0.80 | 0.89 | 0.83 | 1.00 | 0.91 |
| 2 | 2014 | 2015 | BoW | 0.95 | 0.80 | 0.87 | 0.83 | 0.96 | 0.89 |
| | | | Doc2vec | 0.99 | 0.98 | 0.98 | 0.98 | 0.99 | 0.98 |
| | | 2016 | BoW | 0.88 | 0.83 | 0.85 | 0.84 | 0.89 | 0.86 |
| | | | Doc2vec | 0.88 | 0.97 | 0.92 | 0.96 | 0.87 | 0.92 |
| | | 2017 | BoW | 0.84 | 0.81 | 0.83 | 0.83 | 0.85 | 0.84 |
| | | | Doc2vec | 0.77 | 0.98 | 0.86 | 0.96 | 0.69 | 0.80 |

Table 5 shows the results of the cross-dataset validation.

**Table 5.** The result of the cross-dataset validation.

| Method | Training data | Test data | Model | NCD (Benign) | | | MTA (Malicious) | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | P | R | F | P | R | F |
| 1 | MTA | D3M | BoW | 0.57 | 0.96 | 0.71 | 0.90 | 0.36 | 0.52 |
| | | | Doc2vec | 0.65 | 0.89 | 0.75 | 0.86 | 0.59 | 0.70 |
| | D3M | MTA | BoW | 0.40 | 0.96 | 0.56 | 0.93 | 0.25 | 0.40 |
| | | | Doc2vec | 0.59 | 0.93 | 0.72 | 0.95 | 0.66 | 0.78 |
| 2 | MTA | D3M | BoW | 0.89 | 0.88 | 0.88 | 0.87 | 0.89 | 0.88 |
| | | | Doc2vec | 0.98 | 0.96 | 0.97 | 0.96 | 0.98 | 0.97 |
| | D3M | MTA | BoW | 0.74 | 0.98 | 0.84 | 0.97 | 0.67 | 0.79 |
| | | | Doc2vec | 0.79 | 0.99 | 0.88 | 0.99 | 0.74 | 0.85 |

BoW was not effective at all with the method 1. Furthermore, even Doc2vec was not effective enough. Therefore, BoW is not effective in the other environment at all. Besides, it is difficult to detect unseen malicious traffic from transport layer information in the other environment.

Doc2vec was generally more effective than BoW with the method 2. In the case that we used MTA for training data, the best F-measure has reached 0.97. This result means that MTA is superior to D3M as a training data. Moreover, using Doc2vec is effective to detect unseen malicious traffic from application layer information in the other environment.

## 6 Discussion

### 6.1 Accuracy

As the results of the experiments, using Doc2vec in an application layer was the most effective. In the timeline analysis, the best F-measure achieves 0.98 in the next year. The F-measure achieves 0.80 even in 3 years. In the cross-dataset validation, the best F-measure achieves 0.97. Thus, our method is precise, if we can obtain good training data.

### 6.2 Mechanism

In a transport layer, word frequency was the most distinctive feature. The most frequent words were traffic sizes. In a sense, this is rote memorization. In fact, BoW was not effective in the other environment at all. Furthermore, even Doc2vec was not effective enough. Therefore, we concluded that it was difficult to detect unseen malicious traffic from transport layer information.

In an application layer, Doc2vec captured distinctive features other than word frequency. In fact, Doc2vec was also effective in the other environment. Needless to say, word frequency is a fundamental element in a linguistic approach. However, unique word count in network traffic is unrestricted. Representing unrestricted traffic with only word frequency has serious limitations. Doc2vec is based on the distributional hypothesis that words occurring in similar context tend to have similar meanings. Doc2vec enables to calculate the semantic similarity between two traffic and infer similar traffic semantically. We believe network traffic in an application layer has the context, and is like a natural language. Hence, we concluded that our method could detect unseen malicious traffic.

### 6.3 Adaptability

Our method can detect a variety of malicious traffic in the same method. All we have to do is input malicious and benign pcap files. Our method can detect malicious traffic regardless of the attack techniques. No prior knowledge of the attack techniques is required. Our method is available in every protocols which TShark can analyze in an application layer. If attackers change the attack techniques or protocols, our methods learn the characteristic automatically. Besides our method does not demand devising feature vectors. Hence, our method is adaptable to many attack techniques.

### 6.4 Durability

Our method learns the difference of 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. Therefore, 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.

### 6.5 Practical Use

The proposed method was effective in the other dataset. This means the proposed method is powerful and versatile. In this paper, we used malicious and benign pcap files. We can obtain these malicious pcap files easily from the websites, which disclose malicious traffic data. We can also obtain benign pcap files easily from everywhere. Thus, the proposed method has a few constraints in practical use.

## 7 Conclusion

In this paper, we proposed how to construct a corpus from network packets and a generic detection method, which does not depend on attack methods and does not demand devising feature vectors. This paper conducted timeline analysis with MTA dataset which contains captured traffic from EKs between 2014 and 2017. This paper also demonstrated cross-dataset validation which uses MTA dataset and D3M dataset. Consequently, the proposed method can detect up-to-date traffic from EKs. We verified that the proposed method was effective over three years, and effective on the other dataset too. The proposed method achieved the F-measure of 0.98 in the timeline analysis and the F-measure of 0.97 on the other dataset. This result means that using Paragraph Vector is effective on unseen traffic in a linguistic approach.

In this paper, we presumed network packets 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 enable to classify the detail automatically.

**Acknowledgment.** This work was supported by JSPS KAKENHI Grant Number 17K06455.

## References

1. Malware-Traffic-Analysis.net. http://www.malware-traffic-analysis.net/
2. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
3. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: Proceedings of 31st International Conference on Machine Learning, pp. 1188–1196 (2014)
4. Wang, K., Stolfo, S.J.: Anomalous payload-based network intrusion detection. In: Jonsson, E., Valdes, A., Almgren, M. (eds.) RAID 2004. LNCS, vol. 3224, pp. 203–222. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30143-1_11
5. Moore, D., Shannon, C., Brown, D.J., Voelker, G.M., Savage, S.: Inferring internet denial-of-service activity. ACM Trans. Comput. Syst. **24**(2), 115–139 (2006)
6. Bailey, M., Oberheide, J., Andersen, J., Mao, Z.M., Jahanian, F., Nazario, J.: Automated classification and analysis of internet malware. In: Kruegel, C., Lippmann, R., Clark, A. (eds.) RAID 2007. LNCS, vol. 4637, pp. 178–197. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74320-0_10
7. Song, H., Turner, J.: Toward advocacy-free evaluation of packet classification algorithms. IEEE Trans. Comput. **605**, 723–733 (2011)
8. Karagiannis, T., Papagiannaki, K., Faloutsos, M.: BLINC: multilevel traffic classification in the dark. In: Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, pp. 229–240 (2005)
9. Gu, G., Perdisci, R., Zhang, J., Lee, W.: BotMiner: clustering analysis of network traffic for protocol and structure independent botnet detection. In: Proceedings of USENIX Security Symposium, vol. 5, pp. 139–154 (2008)
10. Bilge, L., Balzarotti, D., Robertson, W., Kirda, E., Kruegel, C.: Disclosure: detecting botnet command and control servers through large-scale netflow analysis. In: Proceedings of the 28th Annual Computer Security Applications Conference, pp. 129–138 (2012)
11. Antonakakis, M., Perdisci, R., Dagon, D., Lee, W., Feamster, N.: Building a dynamic reputation system for DNS. In: Proceedings of the 19th USENIX Security Symposium (2010)

12. Antonakakis, M., Perdisci, R., Lee, W., Vasiloglou II, N., Dagon, D.: Detecting malware domains at the upper DNS hierarchy. In: Proceedings of 20th USENIX Security Symposium (2011)
13. Antonakakis, M., Perdisci, R., Nadji, Y., Vasiloglou, N., Abu-Nimeh, S., Lee, W., Dagon, D.: From throw-away traffic to bots: detecting the rise of DGA-based malware. In: Proceedings of 21th USENIX Security Symposium (2012)
14. Rahbarinia, B., Perdisci, R., Antonakakis, M.: Segugio: efficient behavior-based tracking of new malware-control domains in large ISP networks. In: Proceedings of the 2015 IEEE/IFIP International Conference on Dependable Systems and Networks (2015)
15. Kruegel, C., Vigna, G.: Anomaly detection of web-based attacks. In: Proceedings of the 10th ACM Conference on Computer and Communications Security, pp. 251–261 (2003)
16. Choi, H., Zhu, B.B., Lee, H.: Detecting malicious web links and identifying their attack types. In: Proceedings of the 2nd USENIX Conference on Web Application Development, pp. 1–11 (2011)
17. Ma, J., Saul, L.K., Savage, S., Voelker, G.M.: Learning to detect malicious URLs. In: ACM Transactions on Intelligent Systems and Technology, vol. 23, Article no. 30 (2011)
18. Zhao, P., Hoi, S.C.: Cost-sensitive online active learning with application to malicious URL detection. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 919–927 (2013)
19. Invernizzi, L., Miskovic, S., Torres, R., Saha, S., Lee, S., Mellia, M., Kruegel, C., Vigna, G.: Nazca: detecting malware distribution in large-scale networks. In: Proceedings of the Network and Distributed System Security Symposium (2014)
20. Nelms, T., Perdisci, R., Antonakakis, M., Ahamad, M.: Webwitness: investigating, categorizing, and mitigating malware download paths. In: Proceedings of the 24th USENIX Security Symposium, pp. 1025–1040 (2015)
21. Bartos, K., Sofka, M.: Optimized invariant representation of network traffic for detecting unseen malware variants. In: Proceedings of the 25th USENIX Security Symposium, pp. 806–822 (2016)
22. Shibahara, T., Yamanishi, K., Takata, Y., Chiba, D., Akiyama, M., Yagi, T., Ohsita, Y., Murata, M.: Malicious URL sequence detection using event de-noising convolutional neural network. In: Proceedings of the IEEE ICC 2017 Communication and Information Systems Security Symposium (2017)
23. Takata, Y., Akiyama, M., Yagi, Y., Hariu, T., Goto, G.: MineSpider: extracting URLs from environment-dependent drive-by download attack. In: Proceedings of the 2015 IEEE 39th Annual Computer Software and Applications Conference, vol. 2, pp. 444–449 (2015)
24. Jodavi, M., Abadi, M., Parhizkar, E.: DbDHunter: an ensemble-based anomaly detection approach to detect drive-by download attacks. In: Proceedings of the 2015 5th International Conference on Computer and Knowledge Engineering (ICCKE), pp. 273–278 (2015)
25. tshark - Dump and analyze network traffic. https://www.wireshark.org/docs/man-pages/tshark.html
26. gensim. https://radimrehurek.com/gensim/
27. scikit-learn. http://scikit-learn.org/
28. Hatada, M., Akiyama, M., Matsuki, T., Kasama, T.: Empowering anti-malware research in Japan by sharing the MWS datasets. J. Inf. Process. **23**(5), 579–588 (2015)