DeepHTTP: Anomalous HTTP Traffic Detection and Malicious Pattern Mining Based on Deep Learning

Yuqi Yu¹(✉), Hanbing Yan¹(✉), Yuan Ma³, Hao Zhou¹, and Hongchao Guan²

¹ National Computer Network Emergency Response Technical Team/Coordination Center of China, Chaoyang District, Beijing, China

{yyq, yhb}@cert.org.cn

² Beijing University of Posts and Telecommunications, Haidian District, Beijing, China

³ Chongqing Municipal Public Security Bureau, Yuzhong District Branch, Chongqing, China

Abstract. Hypertext Transfer Protocol (HTTP) accounts for a large portion of Internet application-layer traffic. Since the payload of HTTP traffic can record website status and user request information, many studies use HTTP protocol traffic for web application attack detection. In this work, we propose DeepHTTP, an HTTP traffic detection framework based on deep learning. Unlike previous studies, this framework not only performs malicious traffic detection but also uses the deep learning model to mine malicious fields of the traffic payload. The detection model is called AT-Bi-LSTM, which is based on Bidirectional Long Short-Term Memory (Bi-LSTM) with attention mechanism. The attention mechanism can improve the discriminative ability and make the result interpretable. To enhance the generalization ability of the model, this paper proposes a novel feature extraction method. Experiments show that DeepHTTP has an excellent performance in malicious traffic discrimination and pattern mining.

Keywords: Bi-LSTM · Attention mechanism · Anomalous HTTP traffic detection · Malicious pattern mining

1 Introduction

According to the 2018 Internet Security Report released by China National Computer Network Emergency Response Technical Team/Coordination Center (CNCERT/CC) [1], website attacks and exploits occur frequently. How to improve the ability of web attack detection is one of the urgent problems in the field of network security.

Among various network protocols, Hypertext Transfer Protocol (HTTP) occupies a considerable proportion of the application layer traffic of the Internet. Since HTTP traffic can record website access states and request content, it provides an excellent source of information for web application attack detection [2–4]. We focus on HTTP traffic mainly for three reasons. 1) Although protocol HTTPS is used by 57.4% of all the websites [5], HTTP traffic still accounts for a large proportion of network traffic. Research [6] shows that for smaller B2B websites, the uptake of HTTPS is low. Because they lack awareness

of the streaming importance of SSL. Also, the perceived complexity of switching to HTTPS is high. 2) A large majority of malware uses HTTP to communicate with their C&C server or to steal data. Many web application attacks use HTTP, such as Cross-site scripting attack (XSS), SQL injection, and so on. 3) The HTTP protocol is transmitted in clear text, which makes it easier to analyze network behaviors.

In this paper, we design DeepHTTP, a complete framework for detecting malicious HTTP traffic based on deep learning. The main contributions are as follows.

Firstly, unlike researches that only detect malicious URLs (Uniform Resource Locators) [7, 8], we extract both URL and POST body (if the HTTP method is POST) to detect web application attacks. This is of great help to portray network behavior more comprehensively.

Secondly, we perform an in-depth analysis of the types and encoding forms of HTTP traffic requests, then propose an effective method to extract content and structure features from HTTP payload (in this paper, “payload” refers to URL and POST body). Content and structure features are used for classification.

Thirdly, the detection model AT-Bi-LSTM is Bidirectional Long Short-Term Memory (Bi-LSTM) [9] with attention mechanism [10]. Since each HTTP request follows the protocol specification and grammar standards, we treat elements in traffic payload as vocabulary in natural language processing and use Bi-LSTM to learn the contextual relationship. The attention mechanism can automatically dig out critical parts, which can enhance the detection capabilities of the model. Due to the introduction of attention mechanism, the model is more interpretable than other deep learning models.

Finally, we design a module for malicious pattern mining. The “malicious pattern” is essentially a collection of strings representing web attacks. Specifically, we cluster malicious traffic entries and perform pattern mining for each cluster. Then we can generate new rules based on the mined malicious patterns. New rules will be configured into detection systems to capture specific types of web attacks.

In a word, DeepHTTP is a complete framework that can automatically distinguish malicious traffic and perform pattern mining. We set up a process that can verify and update data efficiently. The model is updated periodically so that it can adapt to new malicious traffic that appears over time.

The rest of this paper is organized as follows. Section 2 gives a summary of the relevant research. Section 3 briefly introduces the system framework and data preprocessing methods. The proposed model is introduced in detail in Sect. 4, including the malicious traffic detection model and pattern mining method. We launched a comprehensive experiment to demonstrate the effectiveness of the model. The experimental results are shown in Sect. 5. Section 6 gives the conclusions and future works.

2 Related Work

2.1 Malicious Traffic Detection

In recent years, quite a few researches are aiming for detecting anomaly traffic and web application attacks. Communication traffic contains lots of information that can be used to mine anomaly behaviors. Lakhina et al. [58] perform a method that fuses

information from flow measurements taken throughout a network. Wang et al. [59] propose Anagram, a content anomaly detector that models a mixture of high-order n-grams designed to detect anomalous and “suspicious” network packet payloads. To select the important features from huge feature spaces, Zseby et al. [60] propose a multi-stage feature selection method using filters and stepwise regression wrappers to deal with feature selection problem for anomaly detection. The methods mentioned above care less about the structural features of communication payloads which are important for distinguishing anomaly attacking behaviors and mining anomaly patterns. In this paper, we put forward a structure extraction approach, which can help enhance the ability to detect anomaly traffic. The structure feature also makes an important role in pattern mining.

Existing approaches for anomalous HTTP traffic detection can be roughly divided into two categories according to data type: feature distribution-based methods [11, 12] and content-based methods [13]. Content-based methods can get rid of the dependency of artificial feature extraction and is suitable for different application scenarios. Nelms T et al. [14] use HTTP headers to generate control protocol templates including URL path, user-agent, parameter names, etc. Because Uniform Resource Locator (URL) is rich in information and often used by attackers to pass abnormal information, identifying malicious URLs is a hot studied problem in the security detection [8, 15, 16]. In this paper, we use both URL and POST body (if the HTTP method is POST) to detect web attacks. We do not use other parameters in the HTTP header because these fields (like Date, Host, and User-agent, etc.) have different value types and less valid information.

Various methods have been used for detection. Juvonen and Sipola [18] propose a framework to find abnormal behaviors from HTTP server logs based on dimensionality reduction. Researchers compare random projection, principal component analysis, and diffusion map for anomaly detection. Ringberg et al. [19] propose a nonparametric hidden Markov model with explicit state duration, which is applied to cluster and scout the HTTP-session processes. This approach analyses the HTTP traffic by session scale, not the specific traffic entries. Additionally, there are also many kinds of research based on traditional methods such as IDS (intrusion detection system and other rule-based systems) [3, 20–22]. Since malicious traffic detection is essentially an imbalanced classification problem, many studies propose anomaly-based detection approaches that generate models merely from the benign network data [17]. However, in practical applications, the anomaly-based detection model usually has a high false-positive rate. This problem undoubtedly increases the workload of manual verification.

With the rapid development of artificial intelligence, deep learning has been widely used in various fields and has a remarkable effect on natural language processing. Recently, deep learning has been applied to anomaly detection [8, 23–25]. Erfani et al. [25] present a hybrid model where an unsupervised DBN is trained to extract generic underlying features, and a one-class SVM is trained from the features learned by the DBN. LSTM model is used for anomaly detection and diagnosis from System Logs [24]. In this article, we use deep learning methods to build detection models to enhance detection capabilities.

2.2 Pattern Mining Method

In addition to detecting malicious traffic and attack behaviors, some researches focus on pattern mining of cluster traffic. Most existing methods for traffic pattern recognition and mining are based on clustering algorithms [26, 27]. Le et al. [27] propose a framework for collective anomaly detection using a partition clustering technique to detect anomalies based on an empirical analysis of an attack's characteristics. Since the information theoretic co-clustering algorithm is advantageous over regular clustering for creating a more fine-grained representation of the data, Mohiuddin Ahmed et al. [28] extend the co-clustering algorithm by incorporating the ability to handle categorical attributes which augments the detection accuracy of DoS attacks. In addition to the clustering algorithm, JT Ren [29] conducts research on network-level traffic pattern recognition and uses PCA and SVM for feature extraction and classification. I. Paredes-Oliva et al. [30] build a system based on an elegant combination of frequent item-set mining with decision tree learning to detect anomalies.

The signature generation has been researched for years and has been applied to protocol identification and malware detection. FIRMA [31] is a tool that can cluster network traffic clusters obtained by executing unlabeled malware binaries and generate a signature for each cluster. Terry Nelms et al. [32] propose ExecScent, a system that can discover new C&C domains by building adaptive templates. It generates a control protocol template (CPT) for each cluster and calculates the matching score to find similar malware. These tools have proven to automatically generate valid signatures, but the process still needs to define the composition of the initial signature or template in advance. As far as we know, signature generation is rarely used in web attack detection. The study of pattern mining for malicious traffic is not yet mature.

In recent years, the attention-based neural network model has become a research hotspot in deep learning, which is widely used in image processing [33], speech recognition [34], and healthcare [35]. Attention mechanism has also proved to be extremely effective. Luong et al. [36] first design two novel types of attention-based models for machine translation. Since the attention mechanism can automatically extract important features from raw data, it has been applied to relation Classification [37] and abstract extraction [38]. To the best of our knowledge, as for HTTP traffic detection and pattern mining, proposed models rarely combine sequence models with attention mechanism. Hence, in this paper, we build a model based on attention mechanism, which can get rid of the dependency of artificial extraction features and do well in pattern mining.

3 Preliminaries

3.1 DeepHTTP Architecture and Overview

The “Rule Engine” mentioned in this paper is an engine that consists of many rules. Each rule is essentially a regular expression used to match malicious HTTP traffic that matches a certain pattern. Generally, the expansion of the rule base relies on expert knowledge. It requires high labor costs. And the malicious traffic that the “Rule Engine” can detect is limited. Therefore, we additionally introduce a deep learning model based on the attention mechanism, which can identify malicious traffic entries that are not detected

by the “Rule Engine”. Also, the pattern mining module can automatically extract the string patterns in the traffic payload, which can greatly reduce the workload of rule extraction.

In this paper, rules can be roughly divided into seven categories according to the type of web application attack: File Inclusion (Local File Inclusion and Remote File Inclusion), framework vulnerability (Struts2, CMS, etc.), SQL Injection (Union Select SQL Injection, Error-based SQL Injection, Blind SQL Injection, etc.), Cross-Site Scripting (DOM-based XSS, Reflected XSS, and Stored XSS), WebShell (Big Trojan, Small Trojan and One Word Trojan [39]), Command Execution (CMD) and Information Disclosure (system file and configuration file).

Fig. 1. DeepHTTP architecture.

DeepHTTP is a complete framework that can detect web application attacks quickly and efficiently. In this section, we introduce three stages of DeepHTTP (see Fig. 1), which are training stage, detection stage, and mining stage.

- Training stage.** The core task of this phase is to train the model (AT-Bi-LSTM). It includes data processing and model training. First, we put the labeled dataset into the data processing module to obtain content and structure features of traffic payload. After that, we divide the processed formatted data into training, test, and verification sets and store in the database. To enhance the robustness of the model, we build data sets containing positive and negative samples in different proportions and use cross-validation to train the model.
- Detection stage.** The pre-trained model and the “Rule Engine” are used for anomaly traffic detection. After data processing, new HTTP entries are first entered into the “Rule Engine” for detection. For the entries which are detected by the engine, we labeled the data and update them directly into the database. Other traffic entries will be entered into the pre-trained model for detection. Anomaly traffic entries detected by AT-Bi-LSTM will be used in the mining stage.

- **Mining stage.** The main works of this phase are verifying the anomalous traffic labeled by the model and then mining malicious patterns. Generally speaking, there are a large number of traffic entries that the model identifies as malicious. To improve efficiency, we first cluster and sample the data. Specifically, malicious traffic will be divided into different clusters by clustering. In each cluster, we mine malicious patterns based on attention mechanism and then generate new rules. Simultaneously, we sample a small number of entries from each cluster and perform manual verification. Verified malicious data will be updated regularly to the database and new rules will be updated regularly to “Rule Engine”.

DeepHTTP is a complete closed-loop workflow. The detection model and “Rule Engine” complement each other. The timing update and feedback mechanism can continuously improve the detection ability of the system, which is the main reason for the practicability of the framework. Data processing, traffic detection model, and pattern mining method are critical parts in DeepHTTP, which will describe in the later sections.

3.2 Data Preprocessing

Data Collection. The study spends nearly half a year to collect actual traffic. Nearly 1.5 million malicious HTTP traffic samples are accumulated through vulnerability scanning, rule filtering, and manual verification. After sampling and deduplication, we eventually collect 10, 645, 12 malicious samples.

- **Rule-based collection method.** Specifically, we collect network traffic from the university network monitoring system and filter out HTTP traffic. To protect the privacy of teachers and students, we remove sensitive content from the data. Then, we use the “Rule Engine” mentioned in Sect. 3.1 to identify malicious traffic.
- **Tools-based collection method.** In order to enrich the type of malicious traffic, we use kali [40], Paros [41], W3AF [42] to perform simulation attack and vulnerability scanning. We collect relevant traffic as malicious traffic samples.
- **Model-based collection method.** As described in Sect. 3.1, after manual verification, malicious traffic entries detected by AT-Bi-LSTM are periodically updated to the data set.

Data Cleaning. We parse HTTP traffic packets and extract Uniform Resource Locator (URL) and POST body (if the request method is POST). Then, we mainly perform the following data cleaning operations:

- **URL decoding:** Since URL data often been encoded, we perform URL decoding.
- **Payload decoding:** Many strings in traffic payload are encoded by different encoding methods, like MD5, SHA, and Base64, etc. For these strings, we identify the encoding type and replace them with the predefined flag (see Table 1).
- **We replace garbled characters and invisible characters with null characters.**
- **Since the binary stream data in the Post request body does not contain semantic information, we replace this kind of data with the predefined flag (see Table 1).**

String Segmentation. Text vectorization is the key to text mining. Numerous studies use n-grams [43] to extract the feature of payloads [44–46]. This method can effectively capture the byte frequency distribution and sequence information, but it is easy to cause dimension disaster. To prevent dimensional disaster, we split the string with special characters. The special characters refer to characters other than English letters and numbers, such as “@”, “!”, “#”, “%”, “^”, “&”, “*”, “?”, etc. Here is an instance. Suppose the decoded data is: *“/tienda1/publico/vaciar.jsp <EOS> B2 = Vaciar carrito; DROP TABLE usuarios; SELECT * FROM datos WHERE nombre LIKE”*. “<EOS>” is the connection symbol. After string splitting, the data is denoted as: *“/tienda1 /public /vaciar. jsp <EOS> B2 = Vaciar carrito; DROP TABLE usuarios; SELECT * FROM datos WHERE nombre LIKE”*. Strings are connected by spaces. Another benefit of this approach is that it makes the results of malicious pattern mining more understandable. In this example, the malicious pattern we want to obtain from the data is {“SELECT”, “FROM”, “WHERE”}. However, if we use n-grams ($n = 3$) or character-based method [39], the result may be denoted as {“SEL”, “ELE”, ..., “ERE”} or {“S”, “L”, ..., “R”}, which is not intuitive.

Structure Feature Extraction. To better measure the similarity of URLs, Terry Nirm, etc. [32] use a set of heuristics to detect strings that represent data of a certain type and replaces them accordingly using a placeholder tag containing the data type and string length. Inspired by this, the paper uses a similar way to extract structure features from HTTP payload. The “structure feature” mentioned in this paper refers to string type other than the meaning of the string itself. We replace string with predefined flags according to their data type. The types of data we currently recognize include hash (MD5, SHA, and Base64), hexadecimal, binary, Arabic numerals and English alphabet (upper, lower and mixed case) .etc. The main replacement rules are shown in Table 1.

Table 1. Characters replacement rules.

Encoding type	Replacement string
MD5 hash	‘MD5_HASH’
SHA hash	‘SHA_HASH’
Base64	‘BASE64_ENCODE’
Hexadecimal	‘HEXADECIMAL’
Encryption	‘ENCRYPTION’
Binary	‘BINARY’

Fig. 2. An example of structure extraction.

Here is an example of a structure feature extraction (see Fig. 2). Since the encoding type of the string “3c5fee3560000218bf9c5d7b5d3524e” is MD5 (We use “hashID” [47] to identify the different types of hashes.), we replace it with “MD5_HASH”. For those string not belong to any special type, we replace each character in the string with the specified character. “D” for Arabic numeral and “W” for the English alphabet (not case sensitive). Since the string “68247” consists of five Arabic numerals, we replace it with five “D”. Obviously, by extracting structural features, we can easily find requests with different content but almost the same in data type.

4 Our Approach

4.1 Anomaly HTTP Traffic Detection

The goal of the proposed algorithm is to identify anomaly HTTP traffic based on semantics and structure of traffic entries. Figure 3 shows the high-level overview of the proposed model. The model (AT-Bi-LSTM) contains five components: input layer, word embedding layer, Bi-LSTM layer, attention layer and output layer.

Problem Definition. Let $\mathbf{R} = \{R_1, R_2, \dots, R_i, \dots, R_N\}$ be the set of HTTP traffic entries after data processing. For each traffic entry $R_i (i = 1, 2, \dots, N)$, there are two sequences $S_i^1 = \{c_{11}, c_{12}, c_{13}, \dots, c_{1n}\}$ and $S_i^2 = \{c_{21}, c_{22}, c_{23}, \dots, c_{2n}\}$, which respectively represent content sequence and structure sequence. Because structure sequence is derived from content sequence, the length of both sequence is equal to n .

Fig. 3. Model architecture.

Input Layer. In this paper, we use the content and structure sequence after word segmentation as a corpus, and select words that are common in the corpus to build a vocabulary according to term frequency inverse document frequency (TF-IDF) [48]. Then, the unique index is generated for each word in the vocabulary. We convert the word sequences (S_i^1 and S_i^2) to final input vectors ($S_i^{1'}$ and $S_i^{2'}$), which are composed of indexes. The length of input vector is denoted as z , which is a hyper-parameter (the fixed length in this paper is set to 300 because the proportion of sequence length within 300 is 0.8484). The excess part of input sequence is truncated, and the insufficient part is filled with zero. Formally, the sequence of content can be converted to $S_i^{1'} = \{w_{11}, w_{12}, w_{13}, \dots, w_{1z}\}$ and the sequence of structure can be expressed as $S_i^{2'} = \{w_{21}, w_{22}, w_{23}, \dots, w_{2z}\}$. Here is an example. Given a sequence of content: {'/', 'admin', '/', 'caches', '/', 'error_ches', '.', 'php'}. The input vector with fix length can be denoted as [23, 3, 23, 56, 23, 66, 0, 0, ..., 0]. Since the index of 'admin' in vocabulary is 3, the second digit in the vector is 3. And since the length of this sequence is less than fixed length, the rest of the vector is filled with zeros.

Embedding Layer. Take a content sequence of i -th traffic entry as an example. Given $S_i^{1'} = \{w_{11}, w_{12}, \dots, w_{1k}, \dots, w_{1z}\}$, we can obtain vector representation $v_{1k} \in R^m$ of each word $w_{1k} \in R^1 (k = 1, 2, \dots, z)$ as follows:

$$v_{1k} = \text{ReLU}(W_e w_{1k} + b_e) \quad (1)$$

where m is the size of embedding dimension, $W_e \in R^{m \times 1}$ is the weight matrix, and $b_e \in R^m$ is the bias vector. Rectified Linear Unit (ReLU) is the rectified linear unit defined as $\text{ReLU}(v) = \max(v, 0)$, where $\max()$ applies element-wise to vector.

Bidirectional Long Short-Term Memory. We employ Bidirectional Long Short-Term Memory (Bi-LSTM), which can exploit information both from the past and the future to improve the prediction performance and learn the complex patterns in HTTP requests better. A Bi-LSTM consists of a forward and backward LSTM. Given embedding vector $\{v_{11}, v_{12}, \dots, v_{1k}, \dots, v_{1z}\}$ of content sequence of i -th traffic entry R_i , the forward LSTM f reads the input sequence from v_{11} to v_{1z} , and calculates a sequence of forward hidden states $(\vec{h}_{11}, \vec{h}_{12}, \dots, \vec{h}_{1k}, \dots, \vec{h}_{1z})$ ($\vec{h}_{1k} \in R^p$) and p is the dimensionality of hidden states). The backward LSTM $\bar{f}$ reads the input sequence in the reverse order and product a sequence of backward hidden states $(\bar{h}_{11}, \bar{h}_{12}, \dots, \bar{h}_{1k}, \dots, \bar{h}_{1z})$ ($\bar{h}_{1k} \in R^p$). The final latent vector representation $h_{1k} = \begin{bmatrix} \vec{h}_{1k}; \bar{h}_{1k} \end{bmatrix}^T$ ($h_{1k} \in R^{2p}$) can be obtained by concatenating the forward hidden state $\vec{h}_{1k}$ and the backward one $\bar{h}_{1k}$. We deal with the embedding vector of structure sequence in the same way.

Attention Layer. In this layer, we apply attention mechanism to capture significant information, which is critical for prediction. General attention is used to capture the relationship between h_t and $h_i (1 \leq i < t)$:

$$\alpha_{ti} = h_t^T W_a h_i \quad (2)$$

$$\alpha_t = \text{softmax}([\alpha_{t1}, \alpha_{t2}, \dots, \alpha_{t(t-1)}]) \quad (3)$$

where $W_\alpha \in R^{2p \times 2p}$ is the matrix learned by model, α_t is the attention weight vector calculated by softmax function. Then, the context vector $c_t \in R^{2p}$ can be calculated based on the weights obtained from Eq. (3). The hidden states from h_1 to h_{t-1} can be calculated by the following formulas:

$$c_t = \sum_i^{t-1} \alpha_{ti} h_i \quad (4)$$

We combine current hidden state h_t and context vector c_t to generate the attentional hidden state as follows:

$$\tilde{h}_t = \tanh(W_c[c_t; h_t]) \quad (5)$$

where $W_c \in R^{r \times 4p}$ is the weight matrix in attention layer, and r is the dimensionality of attention state. $\tilde{h}_1$ and $\tilde{h}_2$ can be obtained using Eq. (2) to Eq. (5), which denote the attention vector of content and structure sequence learned by the model.

Output Layer. Before feeding the attention vector into softmax function, the paper apply dropout regularization randomly disables some portion of attention state to avoid overfitting. It is worth noting that we concatenate vector of content and structure to generate output vector for prediction. The classification probability is calculated as follows:

$$p = \text{softmax}(w_s[h_1^*; h_2^*] + b_s) \quad (6)$$

where h_1^* is the output of $\tilde{h}_1$ after dropout strategy, h_2^* is the output of $\tilde{h}_2$. $w_s \in R^{q \times r}$ and $b_s \in R^q$ are the parameters to be learned.

$$\hat{y} = \text{argmax}(p) \quad (7)$$

where $\hat{y}$ is the label predicted by the attention model.

Objective Function. The paper calculate the loss for all HTTP traffic entries using the cross-entropy between the ground truth $y_i \in (0, 1)$ and the predicted $p_i (i = 1, 2, \dots, N)$:

$$L = -\frac{1}{N} \sum_{i=1}^N y_i \log(p_{i1}) + (1 - y_i) \log(1 - p_{i1}) \quad (8)$$

where N is the number of traffic entries, p_{i1} denotes the probability that the i -th sample is predicted to be malicious.

We train the model to minimize the objective function so that the model automatically learns the appropriate parameters. The model can automatically learn the feature expression of input data without manual feature extraction. In addition to outputting the judgment results, the model will also output attention weights which will be used as important inputs for the pattern mining part. The introduction of attention mechanism makes this model more explanatory than other deep learning models.

4.2 Mining Stage

The function of this module is to interpret the results of the model and extract the string pattern. For malicious traffic that is not detected by the rules engine but is discriminated by the model, we perform pattern mining and verification. Figure 4 shows the architecture of the mining stage.

Fig. 4. The architecture of mining stage.

Clustering. We cluster traffic entries that were flagged as malicious by AT-Bi-LSTM. Specifically, we feed the attentional hidden state (obtained by Eq. (5) in Sect. 4.1) into the clustering model. The clustering method we apply is DBSCAN [49], a density-based clustering algorithm, which does not require prior declaring the number of clusters. After clustering, we obtain several clusters. Traffic entries in each cluster are similar in content or structure.

Tag Verification. In practical applications, there are massive suspicious HTTP requests every day. There is no doubt that manual verification requires a lot of time and effort. In this paper, we use clustering and sampling to reduce the workload. After clustering, we sample some entries from each cluster for verification. If the predicted labels of these samples are consistent with the ground-truth, then all the prediction results in this cluster are considered correct.

Pattern Mining. This module can mine the string pattern of the payload of malicious traffic. Experts generate new rules based on the results of pattern mining, which can reduce the workload of manual extraction. As mentioned in Sect. 3.1, the attention weight vector obtained in the attention layer can reflect the crucial parts of the payload. Therefore, for each malicious traffic entry, we dig out the key parts according to the corresponding attention weight vector. The greater the weight is, the more important the word is.

Specifically, given a cluster with N traffic entries $T = \{t_1, t_2, \dots, t_N\}$, we perform pattern mining according to the following steps:

- **Get a keyword set according to attention weight.** AT-Bi-LSTM can output the attention weight vector (obtained by Eq. (3)). For each traffic entry $t_i (i = 1, 2, \dots, N)$, we get n keywords $K_i = \{k_1, k_2, \dots, k_n\}$ according to its weight vector. The greater the weight, the more important the word is. At last, we can obtain a set of keywords $K = \{K_1, K_2, \dots, K_N\}$ identified by the model.
- **Extracting frequent patterns.** The goal of this step is to unearth words that not only frequently occur in this cluster but also recognized by the model as key parts. We calculate the co-occurrence matrix of keywords in set K . If we discovery several words in keywords set K to appear together frequently, then the combination of these words can represent a malicious pattern. The malicious pattern can be used as an effective basis for security personnel to extract new filtering rules.

5 Evaluation

5.1 Dataset

We use the method mentioned in Sect. 3.2 to build the HTTP traffic dataset. For the collected data, we perform manual verification and tagging. Finally, the total number of labeled data is 2,095,222, half of them are malicious traffic entries. The types and quantities of tagged malicious samples are shown in Table 2. Moreover, we prepare five million unmarked HTTP traffic for model testing.

Table 2. Distribution of malicious traffic entries.

Data type	Number
Deserialization	6014
CMS	5836
File inclusion	46438
SQL injection	463776
Webshell	288050
XSS	127750
Sensitive data exposure	16656
Middleware vulnerability	47614
Struts2 vulnerability	42477
Botnet	19901
Total	1064512

5.2 Validation of Structural Feature Extraction Method

To verify the effectiveness of the structural feature extraction method, we compare the convergence speed and detection ability of the model trained by different features.

We record the loss and accuracy of each iteration of the model and draw the loss curve and the accuracy curve (Fig. 5). To balance the memory usage and model training efficiency, the best batch size is set to 200. As we observe from the figure, the model trained based on content and structural features converge faster. In other words, after fusing structural features, the learning rate has been enhanced, and it can reach the convergence state faster.

Fig. 5. Accuracy curve and loss curve.

Moreover, in unbalanced dataset, we compare the effects of models trained by different features. As shown in Table 3, the model trained based on content and structure features performs better. The reason is that structural features increase the generalization ability of the model.

Table 3. Performance of models trained by different features

Different features	Precision	Recall	F1-score	AUC
Content feature	0.9856	0.8765	0.9278	0.9382
Structure feature	0.9633	0.6643	0.7863	0.8320
Content and structure features	0.9560	0.9608	0.9584	0.9795

5.3 Model Comparison

We use 3-gram [43], TF-IDF [48], Doc2vec [50] and Character_level feature extraction method [8, 39] to obtain the feature vector of the payload. Then, we compared the effects of models between classic machine learning methods and machine learning models, including Support Vector Machine(SVM) [51], Random Forest(RF) [52], eXtreme Gradient Boosting(XGBoost) [53], Convolutional neural networks (CNNs) [54], Recurrent neural networks (RNNs) [55, 56], Long short term memory (LSTM) [57] and the proposed model AT-Bi-LSTM.

Detection in Labeled Dataset. We sample 1.1 million traffic entries from labeled dataset (as described in Sect. 3.2) to build a balanced dataset (550,000 for normal samples and 550,000 for malicious samples). To approximate the actual situation, we also sample 1.1 million traffic entries from labeled dataset to build an unbalanced dataset (1,000,000 for normal samples and 100,000 for malicious samples). Then the data set is divided into training set, test set and verification set according to the ratio of 6:2:2. The evaluation metrics consist of precision, recall, F1-score.

Table 4. Model performance in labeled dataset.

Dataset	Classifier	Precision	Recall	F-score
Balanced dataset	3-gram_TF-IDF_SVM	0.9607	0.9564	0.9585
	3-gram_TF-IDF_RF	0.9518	0.9269	0.9378
	3-gram_TF-IDF_XGBoost	0.9755	0.9683	0.9717
	Doc2vec_SVM	0.9365	0.9201	0.9274
	Doc2vec_RF	0.9646	0.9444	0.9534
	Doc2vec_XGBoost	0.9810	0.9753	0.9781
	Doc2vec_CNN	0.9611	0.9467	0.9538
	Doc2vec_LSTM	0.9765	0.9538	0.9650
	Doc2vec_Bi-LSTM	0.9852	0.9791	0.9821
	Character_Level_CNN	0.9556	0.9461	0.9508
	Character_Level_LSTM	0.9895	0.9847	0.9870
	Character_Level_Bi-LSTM	0.9954	0.9921	0.9937
	AT-Bi-LSTM	0.9979	0.9963	0.9970
Unbalanced dataset	3-gram_TF-IDF_SVM	0.6573	0.5987	0.6266
	3-gram_TF-IDF_RF	0.7036	0.6835	0.6934
	3-gram_TF-IDF_XGBoost	0.7499	0.6937	0.7207
	Doc2vec_SVM	0.7531	0.6111	0.6061
	Doc2vec_RF	0.8212	0.7484	0.7675
	Doc2vec_XGBoost	0.8844	0.8570	0.8683
	Doc2vec_CNN	0.8823	0.7851	0.8308
	Doc2vec_LSTM	0.8921	0.8235	0.8564
	Doc2vec_Bi-LSTM	0.9011	0.8221	0.8597
	Character_Level_CNN	0.9365	0.9342	0.9353
	Character_Level_LSTM	0.9485	0.9456	0.9470
	Character_Level_Bi-LSTM	0.9545	0.9574	0.9559
	AT-Bi-LSTM	0.9661	0.9609	0.9635

We can conclude the following conclusions according to Table 4. First, in the balanced dataset, Doc2vec_XGBoost, Character_Level_Bi-LSTM, and AT-Bi-LSTM perform well. However, in the imbalanced dataset, the detection capabilities of Doc2vec_XGBoost is not as good as deep learning models. Second, although the character-level deep learning models are comparable to AT-Bi-LSTM, the model proposed in this article is superior in interpretability. Finally, AT-Bi-LSTM is superior to all baseline models in almost all metrics. In unbalanced data sets, the superiority of the proposed model is even more pronounced.

At the same time, we record the training time of each model (see Fig. 6). Doc2vec-based deep learning models take more time because using Doc2vec to obtain sentence vectors requires additional training time. Because CNN has faster training speed, the training time of Character_Level_CNN is the least. The training time of AT-Bi-LSTM is at the middle level. It is acceptable in practical application.

Fig. 6. Training time of models.

Detection in Unlabeled Dataset. We conduct comparative experiments using five million unlabeled traffic entries. Rules in “Rule Engine” are derived from expert knowledge so that we use the rules engine to verify the validity of the detection model. The explanation of the assessment indicators is as follows:

N_M . The number of malicious entries detected by the model.

N_RE . The number of malicious entries detected by the “Rule Engine”.

$N_M \cap RE$. The number of malicious entries detected by both the model and the “Rule Engine”.

M_RE . A collection of malicious entries detected by the model but not detected by the “Rule Engine”.

N_TP . The number of true positive samples in the M_RE .

N_FP . The number of false positive samples in the M_RE .

N_TP and N_FP are depend on manual verification.

Rule_Coverage_Rate (RCR) = $N_M \cap RE / N_RE$. It represents the coverage of the model towards the “Rule Engine”.

False_Rate (FR) = N_FP / N_M . It means the false rate of the model.

New_Rate (NR) = N_TP / N_M . It represents the ability of the model to identify malicious traffic outside the scope of the “Rule Engine”.

We adopt the “Rule Engine” to extract malicious entries across the overall unlabeled traffic set. The amount of malicious traffic entries detected by “Rule Engine” (NMT_RE) equals to 217100. The result of model evaluation in the unlabeled dataset is shown in Table 5. According to the value of RCR, Doc2vec_Bi-LSTM, Character_level_CNN and AT-Bi-LSTM can basically cover the detection results of the “Rule Engine”. However, Doc2vec_Bi-LSTM and Character_level_CNN have a higher false rate. Overall, AT-Bi-LSTM is superior to other models.

Table 5. Model results in the unlabeled dataset.

Model	N_M	N_RE	$N_M \cap RE$	N_TP	N_FP	RCR	FR	NR
3-gram_TF-IDF_SVM	231246	217100	98965	38957	93324	0.4558	0.4036	0.1685
3-gram_TF-IDF_RF	234796	217100	102578	39875	92343	0.4725	0.3933	0.1698
3-gram_TF-IDF_XGBoost	265478	217100	119867	48057	97554	0.5521	0.3675	0.1810
Doc2vec_SVM	250164	217100	117687	47895	84582	0.5421	0.3381	0.1915
Doc2vec_RF	302546	217100	116598	48965	136983	0.5371	0.4528	0.1618
Doc2vec_XGBoost	348951	217100	124263	53248	171440	0.5724	0.4913	0.1526
Doc2vec_CNN	458964	217100	169542	91458	197964	0.7809	0.4313	0.1993
Doc2vec_LSTM	486525	217100	189981	90259	206285	0.8751	0.4240	0.1855
Doc2vec_Bi-LSTM	589647	217100	200143	99653	289851	0.9219	0.4916	0.1690
Character_level_CNN	653287	217100	198756	180145	274386	0.9155	0.4200	0.2758
Character_level_LSTM	325648	217100	165478	55641	104529	0.7622	0.3210	0.1709
Character_level_Bi-LSTM	295876	217100	187569	31542	76765	0.8640	0.2594	0.1066
AT-Bi-LSTM	428270	217100	206809	110974	110487	0.9526	0.2580	0.2591

5.4 Malicious Pattern Mining

As mentioned before, one of the highlights of AT-Bi-LSTM is that it can automatically identify the malicious part of each traffic request according to attention weight vector. This is also the difference between this model and the traditional fingerprint extraction methods [14, 31]. As described in Sect. 4.2, we first cluster the malicious entries detected by AT-Bi-LSTM but not detected by the “Rule Engine”, then we perform pattern mining for each cluster.

Given a cluster that consists of several traffic of cross-site scripting attack (see Fig. 7). We can get keywords for each entry according to its attention weight vector.


```
/cgi-bin/wa.exe?SHOWTPL=<script>alert(/openvas-xss-test/)</script>
/webContent/fck/wlkt.htm?xss_test"></textarea></script><script>prompt(42873);</script>
adminDirHand="/></script><script>alert(1);</script>
itemid=1527"></textarea></script><scr<script>ipt>alert(2014)</scr<script>ipt>
/survey/list.jsp?s_id=f65b</textarea></script><a href="//eye.hihop.cn/>webscan</a>
```

Fig. 7. Traffic samples of cross-site scripting attack.

For instance, the first traffic entry in Fig. 7 is “/cgi-bin/wa.exe?<EOS>SHOWTPL=<script> alert(/openvas-xss-test/)</script>”. The visualization of its attention vector is shown in Fig. 8. The color depth corresponds to the attention weight α_t (Eq. 3). The darker the color, the greater the weight value. Obviously, top 10 keywords for this entry are {‘.’, ‘exe’, ‘<’, ‘script’, ‘>’, ‘alert’, ‘)’, ‘/’, ‘openvas’, ‘xss’}. Based on this string pattern, we can generate a rule that identifies such malicious traffic.

Fig. 8. Visualization of attention.

Fig. 9. Visualization of pattern mining.

To further illustrate the performance of the proposed model in malicious pattern mining, we visualize the pattern mining results of this cluster (see Fig. 9). The darker the color of the square is, the more times the words appear together. Hence, the pattern of these traffic can be denoted as { "<", "/", " script ", ">", "textarea", "prompt", "javascript", "alert", "iframe", "src", "href" }.

6 Conclusion

This paper presents DeepHTTP, a general-purpose framework for HTTP traffic anomaly detection and pattern mining based on deep neural networks. We build AT-Bi-LSTM, a deep neural networks model utilizing Bidirectional Long Short-Term Memory (Bi-LSTM), which can enable effective anomaly diagnosis. Besides, we design a novel method that can extract the structural characteristics of HTTP traffic. DeepHTTP learns content feature and structure feature of traffic automatically and unearths critical section of input data. It performs detection at the single traffic level and then performs pattern mining at the cluster level. The intermediate output including attention hidden state and the attentional weight vector can be applied to clustering and pattern mining, respectively. Meanwhile, by incorporating user feedback, DeepHTTP supports database updates and model iteration. Experiments on a large number of HTTP traffic entries have clearly demonstrated the superior effectiveness of DeepHTTP compared with previous methods.

Future works include but are not limited to incorporating other types of deep neural networks into DeepHTTP to test their efficiency. Besides, improving the ability of the model to detect unknown malicious traffic is something we need to further study in the future. With the increasing popularity of encrypted traffic, the detection of encrypted traffic attacks is also our future research direction.

Acknowledgement. This work was supported by the National Key R&D Program China under Grant 2018YFB0804701.

References

1. China Internet Network Security Report. <https://www.cert.org.cn/publish/main/upload/File/2019-year.pdf>
2. Estévez-Tapiador, J.M., García-Teodoro, P., et al.: Measuring normality in HTTP traffic for anomaly-based intrusion detection. *Comput. Netw.* **45**(2), 175–193 (2004)
3. Jamdagni, A., Tan, Z., Nanda, P., He, X., Liu, R.P.: Intrusion detection using GSAD model for HTTP traffic on web services. In: *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, pp. 1193–1197. ACM (2010)
4. Tombini, E., Debar, H., Mé, L., Ducassé, M.: A serial combination of anomaly and misuse IDSes applied to HTTP traffic. In: *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC)*, pp. 428–437. IEEE Computer Society, Washington, DC, USA (2004)
5. w3techs Homepage. <https://w3techs.com/technologies/details/ce-httpsdefault>. Accessed 21 Jan 2020

6. Tony Messer's blog published in Pickaweb. <https://www.pickaweb.co.uk/blog/local-business-seo-stats-chart-and-infographic/>
7. Le, H., et al.: URLNet: learning a URL representation with deep learning for malicious URL detection. arXiv preprint [arXiv:1802.03162](https://arxiv.org/abs/1802.03162) (2018)
8. Saxe, J., Berlin, K.: eXpose: a character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys. arXiv preprint [arXiv:1702.08568](https://arxiv.org/abs/1702.08568) (2017)
9. Sundermeyer, M., Schlüter, R., Ney, H.: LSTM neural networks for language modeling. In: Proceedings of INTERSPEECH, pp. 601–608 (2012)
10. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint [arXiv:1409.0473](https://arxiv.org/abs/1409.0473) (2014)
11. Lakhina, A., Crovella, M., Diot, C.: Mining anomalies using traffic feature distributions. SIGCOMM Comput. Commun. Rev. **35**(4), 217–228 (2005)
12. Samant, A., Adeli, H.: Feature extraction for traffic incident detection using wavelet transform and linear discriminant analysis. Comput.-Aided Civil Infrastruct. Eng. **15**(4), 241–250 (2010)
13. Swarnkar, M., Hubballi, N.: OCPAD: one class Naive Bayes classifier for payload based anomaly detection. Expert Syst. Appl. **64**, 330–339 (2016)
14. Nelms, T., Perdisci, R., Ahamad, M.: Execscent: mining for new C&C domains in live networks with adaptive control protocol templates. Presented as part of the 22nd {USENIX} Security Symposium ({USENIX} Security 2013), pp. 589–604 (2013)
15. Ma, J., Saul, L.K., Savage, S., et al.: Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1245–1254. ACM (2009)
16. Ma, J., Saul, L.K., Savage, S., et al.: Identifying suspicious URLs: an application of large-scale online learning. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 681–688. ACM (2009)
17. Bortolameotti, R., van Ede, T., et al.: DECANter: DETECTION of anomalous outbounD HTTP TRaffic by passive application fingerprinting. In: Proceedings of the 33rd Annual Computer Security Applications Conference (ACSAC 2017), pp. 373–386. ACM, New York (2017)
18. Juvonen, A., Sipola, T., Hämäläinen, T.: Online anomaly detection using dimensionality reduction techniques for HTTP log analysis. Comput. Netw. **91**, 46–56 (2015)
19. Ringberg, H., Soule, A., et al.: Sensitivity of PCA for traffic anomaly detection. SIGMETRICS Perform. Eval. Rev. **35**(1), 109–120 (2007)
20. El-Alfy, E.S.M., Al-Obeidat, F.N.: A multicriterion fuzzy classification method with greedy attribute selection for anomaly-based intrusion detection. Procedia Comput. Sci. **34**, 55–62 (2014)
21. Estévez-Tapiador, J.M., et al.: Measuring normality in HTTP traffic for anomaly-based intrusion detection. Comput. Netw. **45**(2), 175–193 (2004)
22. Mahoney, M.V., Chan, P.K.: Learning rules for anomaly detection of hostile network traffic. In: Third IEEE International Conference on Data Mining, pp. 601–604 (2003)
23. Radford, B.J., Apolonio, L.M., et al.: Network Traffic Anomaly Detection Using Recurrent Neural Networks. CoRR abs/1803.10769 (2018)
24. Du, M., Li, F., et al.: DeepLog: anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS 2017), pp. 1285–1298. ACM, New York (2017)
25. Erfani, S.M., et al.: High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. Pattern Recogn. **58**, 121–134 (2016)
26. Chiou, T.-W., Tsai, S.-C., Lin, Y.-B.: Network security management with traffic pattern clustering. Soft Comput. **18**(9), 1757–1770 (2014)

27. Le, T.T., Millaire, A., Asseman, P., De, G.P., ThAČAĪry, C., Ducloux, G.: Novel approach for network traffic pattern analysis using clustering based collective anomaly detection. *Ann. Data Sci.* **2**(1), 111–130 (2015)
28. Ahmed, M., Mahmood, A.N.: Network traffic pattern analysis using improved information theoretic co-clustering based collective anomaly detection. In: Tian, J., Jing, J., Srivatsa, M. (eds.) *SecureComm 2014. LNICST*, vol. 153, pp. 204–219. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23802-9_17
29. Ren, J.T., Ou, X.L., Zhang, Y., Hu, D.C.: Research on network-level traffic pattern recognition. In: *Proceedings of the IEEE 5th International Conference on Intelligent Transportation Systems*, pp. 500–504 (2002)
30. Paredes-Oliva, I., Castell-Uroz, I., Barlet-Ros, P., Dimitropoulos, X., SolAĪ-Pareta, J.: Practical anomaly detection based on classifying frequent traffic patterns. In: *2012 Proceedings IEEE INFOCOM Workshops*, pp. 49–54 (2012)
31. Rafique, M.Z., Caballero, J.: FIRMA: malware clustering and network signature generation with mixed network behaviors. In: Stolfo, S.J., Stavrou, A., Wright, C.V. (eds.) *RAID 2013. LNCS*, vol. 8145, pp. 144–163. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41284-4_8
32. Nelms, T., Perdisci, R., et al.: ExecScent: mining for new C&C domains in live networks with adaptive control protocol templates. In: *Presented as part of the 22nd USENIX Security Symposium (2013)*, pp. 589–604. USENIX, Washington, D.C. (2013)
33. Xu, K., Ba, J., Kiros, R., et al.: Show, attend and tell: neural image caption generation with visual attention. In: *International Conference on Machine Learning*, pp. 2048–2057 (2015)
34. Chorowski, J.K., Bahdanau, D., et al.: Attention-based models for speech recognition. In: *Advances in Neural Information Processing Systems*, pp. 577–585 (2015)
35. Ma, F., Chitta, R., Zhou, J., et al.: Dipole: diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1903–1911. ACM (2017)
36. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015)
37. Zhou, P., Shi, W., Tian, J., et al.: Attention-based bidirectional long short-term memory networks for relation classification. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (vol. 2: Short Papers)*, pp. 207–212 (2016)
38. Ren, P., Chen, Z., Ren, Z., et al.: Leveraging contextual sentence relations for extractive summarization using a neural attention model. In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 95–104 (2017)
39. Zhang, H., et al.: Webshell traffic detection with character-level features based on deep learning. *IEEE Access* **6**, 75268–75277 (2018)
40. Kali Official Website. <https://www.kali.org/>. Accessed 21 Jan 2020
41. Chinotec Technologies Company: Paros - for web application security assessment. <http://www.parosproxy.org/index.shtml>. Accessed 21 Jan 2020
42. Riancho, A.: Web Application Attack and Audit Framework. <http://w3af.sourceforge.net>. Accessed 21 Jan 2020
43. Damashek, M.: Gauging similarity with n-grams: language-independent categorization of text. *Science* **267**(5199), 843–848 (1995)
44. Kloft, M., Brefeld, U., Düessel, P., Gehl, C., Laskov, P.: Automatic feature selection for anomaly detection. In: *Proceedings of the 1st ACM Workshop on Workshop on AISec (AISec 2008)*, pp. 71–76. ACM, New York (2008)
45. 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

46. Zolotukhin, M., Härmä, T., Kokkonen, T., Siltanen, J.: Analysis of HTTP requests for anomaly detection of web attacks. In: 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing, pp. 406–411 (2014)
47. hashID, a tool written in Python 3. <https://github.com/psypana/hashID>
48. Joachims, T.: A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In: Proceedings of International Conference on Machine Learning, pp. 143–151 (1996)
49. Ester, M., Kriegel, H.P., Sander, J., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, vol. 96, no. 34, pp. 226–231 (1996)
50. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International Conference on Machine Learning (2014)
51. Cortes, C., Vapnik, V.: Support-vector networks. *Mach. Learn.* **20**(3), 273–297 (1995)
52. Ho, T.K.: Random decision forests. In: Proceedings of 3rd International Conference on Document Analysis and Recognition, vol. 1. IEEE (1995)
53. Chen, T., Guestrin, C.: Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM (2016)
54. Lawrence, S., Giles, C.L., Tsoi, A.C., Back, A.D.: Face recognition: a convolutional neural-network approach. *IEEE Trans. Neural Netw.* **8**(1), 98–113 (1997)
55. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. *Nature* **323**(6088), 533–536 (1986)
56. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **45**(11), 2673–2681 (1997)
57. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
58. Lakhina, A., Crovella, M., Diot, C.: Characterization of network-wide anomalies in traffic flows. In: Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement (IMC 2004), pp. 201–206. ACM, New York (2004)
59. Wang, K., Parekh, J.J., Stolfo, S.J.: Anagram: a content anomaly detector resistant to mimicry attack. In: Zamboni, D., Kruegel, C. (eds.) RAID 2006. LNCS, vol. 4219, pp. 226–248. Springer, Heidelberg (2006). https://doi.org/10.1007/11856214_12
60. Iglesias, F., Zseby, T.: Analysis of network traffic features for anomaly detection. *Mach. Learn.* **101**(1–3), 59–84 (2015)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.