Detecting Malicious Web Requests Using an Enhanced TextCNN

Lian Yu[a], Lihao Chen[a], Jingtao Dong[a], Mengyuan Li[a], Lijun Liu[b], Bei Zhao[b], Chen Zhang[b]
[a]School of Software and Microelectronics, Peking University
[b]Institute of Research and Development, Mobile China
Beijing, China
lianyu@ss.pku.edu.cn

***Abstract*—This paper proposes an approach that combines a deep learning-based method and a traditional machine learning-based method to efficiently detect malicious requests Web servers received. The first few layers of Convolutional Neural Network for Text Classification (TextCNN) are used to automatically extract powerful semantic features and in the meantime transferable statistical features are defined to boost the detection ability, specifically Web request parameter tampering. The semantic features from TextCNN and transferable statistical features from artificially-designing are grouped together to be fed into Support Vector Machine (SVM), replacing the last layer of TextCNN for classification. To facilitate the understanding of abstract features in form of numerical data in vectors extracted by TextCNN, this paper designs trace-back functions that map max-pooling outputs back to words in Web requests. After investigating the current available datasets for Web attack detection, HTTP Dataset CSIC 2010 is selected to test and verify the proposed approach. Compared with other deep learning models, the experimental results demonstrate that the approach proposed in this paper is competitive with the state-of-the-art.**

Keywords—Web security, Malicious Web requests detection, Transferable statistical features, Deep learning, Convolutional Neural Network for Text Classification (TextCNN), Support vector machine (SVM)

I. INTRODUCTION

The 2016 Data Breach Investigations Report [1][2] shows that Web application attacks increased by 400% from 2015 to 2016. The third quarter of 2017 increased by 69% compared to the third quarter of 2016 [3], and the fourth quarter of 2017 increased by 10% compared to the fourth quarter of 2016 [4]. The consequences of Web attacks can be drastic, and imply huge losses of money or reputation. Therefore, safe-guarding of Web applications and their users is urgent.

Uniform Resource Locators (URLs) are carriers of Web requests to Web applications and play a significant role in the connection between clients and Web servers. It is crucial to detect potential malicious requests carried by URLs that threat the security of Web applications. The existing research on malicious Web requests detection still has the following issues:

- Each method has both advantages and disadvantages. Rule-based detection method can detect known attacks precisely by matching abnormal patterns, but is incapable of identifying unknown attacks; traditional machine learning-based detection method is feasible to detect new attacks, but needs artificial feature engineering; deep learning-based detection method can learn abstract features in an automatic approach with intensive computation requirements.
- Deep learning-based detection method is poorly interpretable. It is usually taken as a black-box and not understandable for domain researchers. For example, Recurrent Neural Network (RNN) was used to detect Web attacks in [8]. However, the meaning of intermediate vector in RNN cannot be accessible for human to understand. In other words, it is difficult to interpret what information is vital for detecting malicious URL requests. In addition, it is difficult to adjust the hyper-parameters of deep learning models.
- There are only a few public datasets for Web attack detection analysis on the Internet. Compared with KDD CUP 1999 Dataset [32], HTTP Dataset CSIC 2010 [5] is more suitable for Web attack analysis. However, the dataset has a large portion of duplicate data, which leads to fake high detection accuracy of machine learning models. As Web attack messages like data in HTTP Dataset CSIC 2010 often are encoded due to technical standards or attacker's tricks, raw data in the dataset needs proper processing, decoding, before fed into neural network, otherwise, it is probably not practicable to learn useful features from those encoded texts.

The key contributions of this paper can be summarized as follows:

- Extracting features both using deep learning approach and statistical approach: TextCNN [20] is used for extracting automatically features, and transferable statistical features are defined to enhance the feature representation, which are transferable to other relevant domains. Section V describes the details.
- Enhancing TextCNN model: This paper replaces the last layer in TextCNN with SVM. The first a few layers in TextCNN are utilized for auto-extracting abstract features, and the last SVM-layer is for classifying Web requests. The experiments in Section VI shows that the proposed classifier is competitive with the state-of-the-art on the CSIC dataset in terms of accuracy and recall.

This work is supported by Ministry of Education-China Mobile (MCM20170406) and the National Natural Science Foundation of China (Grant No. 61872011).

978-1-7281-7303-0/20/$31.00 ©2020 IEEE
DOI 10.1109/COMPSAC48688.2020.0-167

- Rolling out the abstract features extracted by TextCNN: The abstract features from TextCNN are vectors with numeric symbols inside, which are obscure and not intuitive for domain researchers to analyze and validate the model. This paper defines tracing-back functions from max-pooling operation back to convolution operations in TextCNN, with the abstract features or vectors as inputs and URL words as outputs. The words correspond to features that are human understandable, called words-of-interest (WOI) in this paper, which is discussed in Section VII.
- Pre-processing the CSIC dataset: The portion of duplication data in the dataset is up to 81%. Experiments show that duplication can lead to fake high accuracy of detection that indeed happened in existing research, which exaggerates the capability of the prediction model. In addition, Web attackers utilize encoding to hide attack intention, and the raw data in the dataset are decoded before fed into training model to improve the training performance as described in Section IV.

The rest of this paper is organized as follows. Section II presents related work on datasets related to malicious Web requests detection and the methodologies; Section III describes the overall process of the proposed approach; Section IV depicts the pre-processing on the CSIC dataset; Section V illustrates the proposed modelling; Section VI shows the experiments on the CSIC dataset; Section VII further discusses the related questions, including duplication in dataset, and finally, Section VIII concludes the paper.

II. Related Work

This section presents datasets and machine learning approaches related to malicious Web requests detection.

A. Datasets

Three available datasets in public for attack detection are DARPA KDD Cup 1999 dataset, ECML/PKDD 2007 dataset [33] and HTTP Dataset CSIC 2010.

Knowledge Discovery in Database (KDD) Cup 1999 has four types of anomalies in the dataset: Denial of Service (DOS), Remote-to-Login (R2L), User-to-Root (U2R) and Probing, which have 39 different attack types, such as smurf, port sweep, buffer overflow, spy, ipsweep, in the test dataset. ECML-PKDD 2007 has evolved from two separate conference series: the European Conference on Machine Learning (ECML) and the European Symposium on Principles of Knowledge Discovery and Data Mining (PKDD), described in Extensible Markup Language (XML). The Dataset contains different kinds of attacks, such as Cross Site Scripting (XSS), SQL Injection, LDAP Injection, XPATH Injection, Path traversal, Command Execution and SSI.

The two datasets, KDD Cup 1999 and ECML-PKDD 2007, are built for competition, and the attack requests in the datasets are blindly constructed [33], which does not involve real Web applications. The aging of datasets and the lack of standardization have become a major problem in the field of intrusion detection.

HTTP Dataset CSIC 2010 used in this paper is developed by Information Security Research Institute of Spanish Research National Council (CSIC). The dataset contains the generated traffic targeted to an e-Commerce Web application based on real data, with more than 72,000 normal requests and 25,000 exception requests including XSS, SQL injection, CRLF (Carriage Return Line Feed) injection, and buffer overflow.

B. Techniques for Detection

1) Traditional Machine Learning Approaches: Researchers have used many methods in experiments to detect Web application attacks for a long time. Some of them use traditional machine learning algorithms for attack detection on the CSIC dataset. In their experiments, various methods are usually used to extract cumbersome features, and then to classify inputs by different classifiers.

The common algorithms include C4.5, CART, and random forest. To filter data in Web application firewalls (WAFs), Nguyen et al. [13] used Correlation Feature Selection (CFS) and minimal-Redundancy-Maximal-Relevance (mRMR) to extract features from data, and used C4.5/CART/Random Forest to detect Web attacks. The effects of each model after feature extraction were compared. CFS and mRMR are two commonly used feature extraction methods.

In addition, some researchers use N-gram model to extract features. Vartouni et al. [12] used N-gram model to construct features for HTTP request data. To reduce the dimension of the problem, Stacked Auto-Encoders (SAEs) with different configurations were used to extract relevant features from the dataset. Zhang et al. [15] used N-gram to extract the words contained in the payload, and used Bayesian reasoning to learn normal and abnormal traffic patterns from the words extracted by training.

Among these methods, feature extraction is an important step. Epp et al. [16] used an One-Class SVM classifier to detect abnormal HTTP requests on the basis of feature extraction of data. The method relies on the previous feature extraction, and it is necessary to construct an effective feature extraction method to improve the detection efficiency. However, previous feature extraction may also lead to unsafe events. Pastrana et al. [14] pointed that an adversary who knows the distribution of datasets and the construction of specific features can generate a targeted attack to avoid classifiers. Especially when the classifier uses a simple feature construction method based on 1-gram, it is easier for the opponent to avoid it.

2) Deep Learning Approaches: The traditional methods mentioned above are facing problems in several aspects. In recent years, deep learning methods have been gradually applied and achieved good results. Wang et al. [11] compared and evaluated CNN (Convolutional Neural Network), LSTM (Long Short-Term Memory) and their combination methods. According to their experimental results, compared with traditional learning methods, deep learning method has obvious advantages. The use of neural networks does solve some difficult problems. Bochem et al. [6] proposed an anomaly detection method for HTTP requests based on LSTM neural network model without domain knowledge. Ito et al. [9] used Character-level Convolutional Neural Network to extract the characteristics of HTTP requests and identify them as normal or abnormal requests.

Zhang et al. [7] improved CNN design and then used it to detect network attacks. In their paper, only simple preprocessing is needed for the original data, and then complex feature extraction is completed by CNN. Liang et al. [8] used RNN (Recurrent Neural Network) to detect network anomalies. They first trained two RNNs to learn the normal request pattern in an unsupervised way, and then trained a MLP (Multi-layer Perceptron). The output of the former two RNNs was used as an input of the MLP to distinguish between normal and abnormal requests, thus eliminating the tedious task of feature extraction. Hao et al. [10] used Bi-directional LSTM (Bi-LSTM) to improve the ability of RNN to get a better performance.

This paper proposes an approach to perform malicious Web requests detection, i.e., the first a few layers in TextCNN are utilized to auto-extract features and the last layer uses SVM as the final classifier. The proposed approach achieved the best results compared with above methods on the same dataset.

III. Overall Process

The overall process of the proposed approach consists of three steps:

- Data pre-processing: It extracts HTTP request data from the raw CSIC dataset, removes duplicated Web requests, converts each character in request data string to a lowercase, decodes characters in URL requests encoded by browsers as normal or by attackers deliberately, and segments each request string into a set of words according to URL syntax or special characters. This part is described in Section IV.A.
- Pre-training TextCNN: Word2vec based on Skip-Gram [22] is used to generate the embedding matrix which can map text words to word vectors. In addition, to have an intuitive understanding of the high-dimensional word vectors of Web requests, the dimensionality of each word vector is reduced and each word vector is finally visualized as a point in a two-dimensional map using t-SNE [23] technique. The embedded matrix can be regarded as the first layer of the entire model and can be tuned during the pre-training process. TextCNN is applied with Softmax loss function. At the end of this step a trained TextCNN model is produced, which is utilized to obtain powerful semantic features from raw data. This part is described in Section IV.B and Section V.A.
- Creating a classifier for TextCNN with inputs of varied features: Excepting for the features from TextCNN, transferable statistical features are extracted, which can make contributions to strengthen the detection ability of malicious URL requests, specifically request parameter tampering. The two kinds of features are grouped and utilized by SVM classifier for training and detecting as well. This part is described in Section V.B and Section V.C.

IV. Data Preprocessing

Compared with other datasets, the CSIC dataset is more suitable regarding the analysis goal in this paper. This section presents data formatting, Web request parsing, and word embedding.

A. Data Formatting

The CSIC dataset contains thousands of Web requests that are automatically generated based on real data and labeled as normal or anomalous. The authors of CSIC dataset originally used it for evaluating Web attack detection methods and systems. All anomalous requests are divided into static attacks that try to access hidden (or non-existent) resources, dynamic attacks, such as SQL injection and XSS, and unintentional illegal requests such as not following the normal Web application behavior or having the normal parameter pattern.

1) Web requests in the CSIC dataset: A typical request in the dataset is mainly composed of a request line and several other lines in the header as shown in Fig. 1. The header information of all requests in the CSIC dataset is exactly the same excepting for the cookie line, and this paper uses the URL of each request line as the payload. A POST request may have a message body. To maintain consistency, for the POST request, the message body is spliced with the URL of request line to serve as a URL payload. Hereafter, the URL payload is simply referred to as a URL, which represents a Web request.

2) URL-decoding: According to RFC2616 and RFC3986, URLs can only be sent over Internet using limited characters, such as English letters and numbers. Other special characters used in the URL will be replaced by "%" followed by two hexadecimal digits. URLs cannot contain the space which is normally replaced by "+". To obtain the semantic information of URLs, it is essential to decode URLs. After decoding the URL once, "%25" is converted to "%", as shown in Fig. 2.

On the other hand, an attacker may perform additional encoding to hide his attack intents. Therefore, decoding may need to be performed several times to completely expose the hidden information. For example, the transition from "%253CSCRi pt%253E" to "<SCRipt>" needs twice decoding, and "<SCRipt>" is exactly an important symbol of the XSS attack as shown in Fig. 2.

```
GET http://localhost:8080/tienda1/publico/autenticar.jsp?modo
=entrar&l ogin=bob%2540%253CSCRipt%253Ealert%2528Paro
s%2529%253C%252FscrIPT%253E.parosproxy.org&pwd=c69p
04e13&remember=off&B1=Entrar HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux)
KHTML/3.5.8 (like Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml+xml,text/ht
ml;q=0.9,text/ plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=9AE932E23A03425BF74EAA9947FEAE3F
Connection: close
```

Fig. 1. A typical request example

URL:
localhost:8080/tienda1/publico/autenticar.jsp?modo=entrar&login=bob%2540%253CSCRipt%253Ealert%2528Paros%2529%253C%252FscrIPT%253E.parosproxy.org&pwd=c69p04e13&remember=off&B1=Entrar
Decode Once:
localhost:8080/tienda1/publico/autenticar.jsp?modo=entrar&login=bob%40%3CSCRipt%3Ealert%28Paros%29%3C%2FscrIPT%3E.parosproxy.org&pwd=c69p04e13&remember=off&B1=Entrar
Decode Twice:
localhost:8080/tienda1/publico/autenticar.jsp?modo=entrar&login=bob@<SCRipt>alert(Paros)</scrIPT>.parosproxy.org&pwd=c69p04e13&remember=off&B1=Entrar

Fig. 2. Decoding the URL

localhost : 8080 localhost : 8080 / tienda1 / publico / autenticar . jsp ? modo = entrar & login = bob @ < script > alert (paros) < / script > . parosproxy . org & pwd = c69p04e13 & remember = off & b1 = entrar

Fig. 3. Converting a URL to lowercase and splitting the URL to individual words

TABLE I. DATA SIZES IN THE CSIC DATASET

Label	Original Size	Distinct Size	Repetition Rate
Normal	72000	9330	87%
Anomalous	25065	8923	64%
All	97065	18253	81%

3) Converting URLs to lowercase and splitting URLs: Attacks may exploit a mixture of uppercase and lowercase like "SCRipt" to bypass detection, and eventually "SCRipt" is treated as a script command to perform harmful actions. To capture the feature, all URLs are converted to lowercases. Some special symbols in a URL have specific syntax meaning, such as "&" jointing parameters, and "=" connecting a pair of key-value in a parameter. Accordingly, this paper separates a URL based on these special symbols and treats separated portions and the symbols as a sequence of words. Fig. 3 shows an example of such a sentence of words separated by spaces.

4) Dealing with duplication of data in the dataset: There are a large portion of duplicated data in the dataset, and if not getting removed, training set and test set will have a lot of overlapped data, which will damage the correctness of the evaluation results (discussed in Section VII.B). Regardless of the request mode (POST, GET or other), data is considered as duplication as long as the URL is the same. Table I presents the original sizes, the distinct sizes and the repetition rate in the dataset regarding normal and anomalous requests. It shows the high rates of repetitions for both categories.

B. Word2vec based on Skip-Gram

To make it possible to process the 'word' by mathematical model, literal characters are converted into word vectors using word embedding technology. After comparing two commonly used word embedding methods, this paper uses skip-gram based Word2vec, and visualizes the generated word vectors using t-SNE.

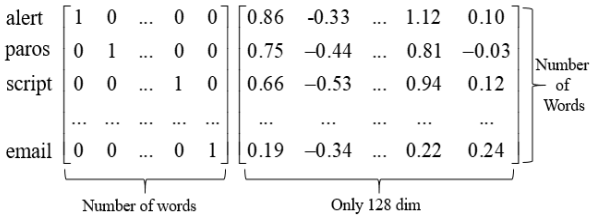

Fig. 4. Word vectors generated by one-hot and Word2vec

TABLE II. COSINE SIMILARITY OF WORD VECTORS

	alert	paros	script	email
alert	1	0.9725	0.9295	0.1782
paros	0.9725	1	0.9310	0.1241
script	0.9295	0.9310	1	0.1270
email	0.1782	0.1241	0.1270	1

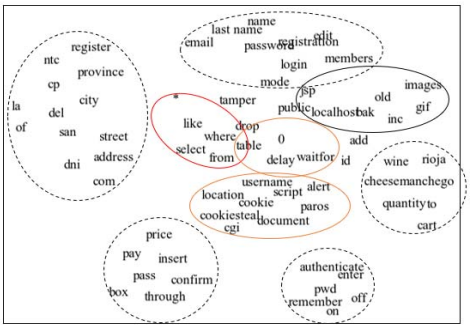

Fig. 5. Words clustering using t-SNE

1) Word embedding: One-hot encoding uses *n*-dimensional binary vectors to encode *n* words, and only one specific dimension of each word vector is 1, where large *n* leads to large word vectors. Word2vec based on skip-gram first establishes a network with a hidden layer to predict the context of a word, and then extracts the parameters of the hidden layer as the word vectors. The number of neurons *k* in the hidden layer is exactly the word vector dimension. Fig. 4 shows word vectors generated by these two methods. Obviously, unlike one-hot code method on the left hand, Word2vec based on skip-gram embeds word vectors into a lower-dimensional semantic space to generate a word vector matrix. This paper uses Word2vec based on skip-gram for word embedding.

Word2vec algorithm fully combines semantic information of the contexts when producing the word vector of a word, i.e., the words that are more closely related in semantics are also closer in the vectors. For example, the cosine similarities among "alert", "paros" (related to a vulnerability scanner) and "script" exceed 0.9, and three words all have low cosine similarity with "email" as shown in Table II.

2) Visualization: To further demonstrate that Word2vec can capture the semantic connection of contexts, t-SNE algorithm is used to embed word vectors from high dimensions into two dimensions and the results of reduction is shown visually in Fig. 5. To show more clearly, words are filtered according to word frequency or whether they can significantly

represent an attack, and then translated from Spanish (some words of raw data in the dataset are Spanish) into English. Words circled in red, orange and black are related to SQL injection, XSS attacks, and file suffixes, respectively. The rest of the words circled by the dotted lines are mostly parameters that often appear together.

To make the word vector representation more specific for the dataset, the embedding matrix generated by Word2vec based on skip-gram is embedded in the network model, and its parameters can be tuned and adjusted by the back-propagation algorithm during the training process.

V. Modelling Malicious Web Requests Detection

This paper models malicious requests detection as a binary classification. The task has three key activities: applying TextCNN for automatic feature extraction; creating transferable statistical features, and building an SVM model in the last layer of TextCNN to perform the classification based on the concatenated features.

A. Applying TextCNN for Feature Extraction

The architecture of TextCNN module is shown in Fig. 6 which has a classical architecture (CONV-ReLU-POOL) [35].

1) Mathematical notation: Different from NLP (nature language processing), a sentence in this paper is a sequence of words obtained by preprocessing a URL representing a Web request, as described in Section IV. Fig. 3 shows an example of such a sentence, where each word is separated by a space. A sentence is formally defined as an ordered set as follows:

$$sentence = \{word_1, \cdots, word_i, \cdots, word_d\} \quad (1)$$

where d is the fixed length (padded or truncated when necessary) of each URL sentence, and $word_i$ is the i-th word.

In addition, the following symbols are defined and used in the next subsection.

$\boldsymbol{x}_i \in \mathbb{R}^k$: A word vector corresponding to $word_i$, where k is the size of a word vector.

$\boldsymbol{x}[i:i+j] \in \mathbb{R}^{(j+1)\times k}$: The concatenation of word vectors $\boldsymbol{x}_i, \boldsymbol{x}_{i+1}, \cdots, \boldsymbol{x}_{i+j}$. $\boldsymbol{x}[1:d] \in \mathbb{R}^{d\times k}$ denotes the sentence matrix by

$$\boldsymbol{x}[1:d] = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_i, \cdots, \boldsymbol{x}_d] \quad (2)$$

$\boldsymbol{w} \in \mathbb{R}^{s\times k}$: A filter of window size s involved in convolution operation. $\boldsymbol{W}$ is the set of all engaged filters and $\boldsymbol{w}_j \in \boldsymbol{W}$ is the j-th filter, $j = 1,2,\ldots,|\boldsymbol{W}|$.

m_i^j: A new feature produced from filter $\boldsymbol{w}_j$ based on word vectors $\boldsymbol{x}[i:i+s-1],\ i = 1,\ldots,d-s+1$.

2) Modeling: Convolution operations are applied on sentence matrix $\boldsymbol{x}[1:d]$. As the filter $\boldsymbol{w}_j$ slides through the sentence matrix, new features are produced. For example, a new feature m_i^j is produced from the window corresponding to word vectors $\boldsymbol{x}[i:i+s-1]$ by

$$m_i^j = \sigma\left(\boldsymbol{w}_j \cdot \boldsymbol{x}[i:i+s-1] + b\right) \quad (3)$$

where σ is a non-linear activation function, such as ReLU (Rectified Linear Unit) [35], and " $\cdot$ " is *dot* product operator and b is a bias term. A feature map is generated from filter $\boldsymbol{w}_j$ and can be represented as

$$\boldsymbol{m}^j = \left[m_1^j, m_2^j, m_3^j, \cdots, m_{d-s+1}^j\right] \quad (4)$$

In the next, the most important feature is extracted from the feature map $\boldsymbol{m}^j$, with a max-pooling operation, and mathematically written as

$$\hat{m}^j = max\{m_i^j \,|\, m_i^j \in \boldsymbol{m}^j\} \quad (5)$$

Another purpose of max-pooling operation is to reduce the dimensionality of the feature map, which leads to a lower computational complexity.

In short, one feature is extracted from one filter, i.e. the filter $\boldsymbol{w}_j$ generates the feature $\hat{m}^j$. To extract multiple different features from the original URL sentence, multiple filters of different window sizes are applied. The output of TextCNN module can be denoted as follows, indicating a set of features automatically extracted:

$$\hat{\boldsymbol{m}} = \left[\hat{m}^1, \hat{m}^2, \hat{m}^3, \cdots, \hat{m}^{|\boldsymbol{W}|}\right] \quad (6)$$

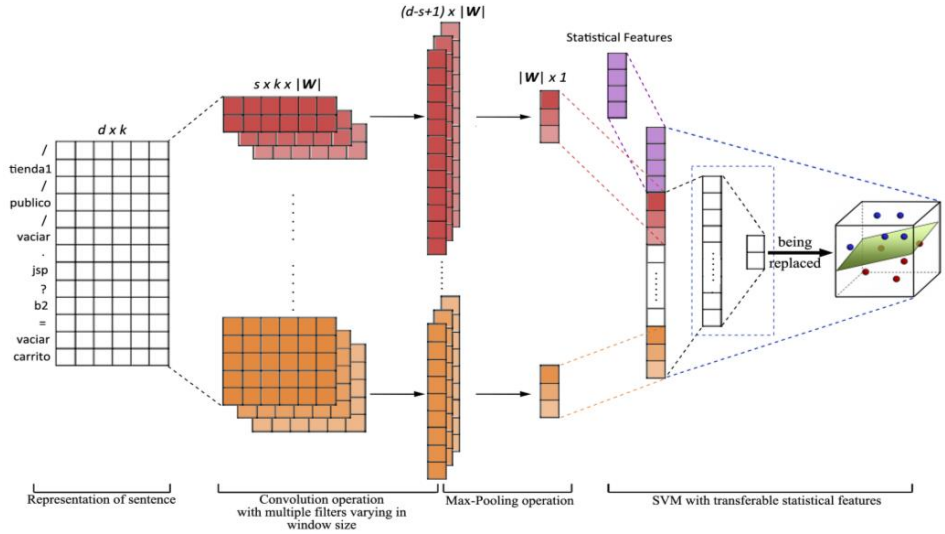

Fig. 6. Architecture of the enhanced TextCNN model

B. Creating Transferable Statistic Features

Statistic features are designed to enhance the performance of the proposed model. As URL format needs to confirm to the requirements of RFC-1738 [34], these statistic features are transferable and easy to be applied to process URLs in other datasets. Specially, this paper focuses on Web security on Web server side, thus raw URL plaintexts can be acquired no matter the application protocol is HTTP or HTTPS.

1) URL-decoding counts: As Fig. 2 shows, Web attackers tend to encode the URL request more than once to wrap up their attacking intention, while there is just only once encoding in normal URL requests. As a result, the times that the URL is encoded can help to detect malicious Web requests. As the URL is decoded, T_{decode} is recorded whether the URL has been encoded more than once.

2) Illegal pattern recognition for key: A URL is composed of one absolute path and several query parameters. Each query parameter has the same format: & key = value &, e.g., & id = 2& where 'id' is the key and '2' is the value. The anomalous data in the CSIC dataset contains the type of parameter tampering where there are many normal parameters like 'provincia', but in some anomalous data, the parameter 'provincia' is modified to 'provinciaa'. To capture this kind of tampered information, a set of normal keys from the training dataset is maintained. For each URL, if there exists a key not in the normal set, edit distance d_{edit} is calculated. If the minimum d_{edit} satisfies the following inequality:

$$d_{edit} < threshold \tag{7}$$

where *threshold* is 2 in the experiment, the key is thought to have been modified. A URL may be anomalous if containing such a tampered key. The number of illegal keys is denoted as $T_{illegal\_key}$.

3) Illegal pattern recognition for value: Parameter tampering involves not only tampering with keys, but also padding illegal values. For example, email=as@@@ter@ ropadegolf.qa is an obvious illegal email address. To catch the tampering of value pattern, a set of value patterns with normal samples in the training set is maintained. The actions are as follows: segmenting each value with special characters, replacing pure numeric by <NUM>, special characters by <SRC>, numbers mixed with normal characters by <STR>, and recording the compounded patterns in the corresponding key's value, which allows the key's value to correspond to several different patterns. Table III shows two examples of the format.

The set of normal value patterns is set up according to the above rules. Next, for every key-value pair in URLs，actions are taken as follows: get the value patterns and compare them with value patterns in the set of normal value patterns corresponding to the key. If not hitting in the set, the value pattern may be anomalous. The times that anomalous value patterns appear is counted as a feature, denoted as $T_{illegal\_value}$.

TABLE III. VALUE PATTERNS' REPLACEMENT

Request Parameter	Request Parameter Pattern
id = 2	id: <NUM>
email = szmidt@surcosur.id	email: <STR><SRC><STR> <SRC><STR>

At last, these transferable statistical features can be grouped and represented as follows:

$$\boldsymbol{t} = [T_{decode}, T_{illegal\_key}, T_{illegal\_value}] \tag{8}$$

C. Embedding SVM in TextCNN

This paper trains an SVM classifier to discriminate between anomalous and normal requests and embeds it in TextCNN at the last layer. The input vector of SVM can be represented as

$$\boldsymbol{c} = [\hat{\boldsymbol{m}}, \boldsymbol{t}] \tag{9}$$

where $\hat{\boldsymbol{m}}$ (Formula (6)) is the feature vector extracted from TextCNN designed in Section V.A, and $\boldsymbol{t}$ (Formula (8)) is the features created in Section V.B. An optimal separating hyper-plane is learned by maximizing geometric margin between support vectors [21], which is what SVM model actually does in binary classification. Let n denote the number of training samples, y_i be the true label (+1 for anomalous request and -1 for normal one) of the i-th sample, $i \in [1, n]$. The output, $\hat{y}_j$ (corresponding to the j-th sample), of SVM can be represented as

$$\hat{y}_j = \mathrm{sgn}\left(\sum_{i=1, i\neq j}^{n} y_i \alpha_i K(\boldsymbol{c}_i, \boldsymbol{c}_j) + b\right), \tag{10}$$

where α_i is a learned parameter, $\boldsymbol{c}_i$ is an input vector corresponding to the i-th sample, b is a bias term and K is a kernel function. The nature of kernel function is to map non-linear data to the high-dimensional space with small computations, and make non-linear data linearly separable in high-dimensional space. Specifically, Gaussian Radial Basis Function (RBF) kernel option is selected in this paper. Eventually the model can predict that a given Web request is normal or anomalous.

Why is SVM used to enhance TextCNN for classification?

- The amount of malicious data used in this paper is relatively small (abnormal data is much fewer than normal data in the realistic world) and SVM performs well on small dataset.
- For given hyper-parameters and a training set, the performance of SVM on the test set is always stable no matter how many times SVM is trained. However, for TextCNN, the performance on the same test set is different for each time TextCNN is trained, which leads to inconsistency on accuracy. As a consequence, the hyper-parameters of TextCNN is harder to be tuned than SVM, and SVM is much more consistent as well.

VI. THE IMPORTANCE OF FEATURES FROM DIFFERENT SOURCES (FEATURES FROM TEXTCNN AND STATISTICAL FEATURES) IS USUALLY DIFFERENT. THE CORRESPONDING WEIGHTS CAN BE INTRODUCED TO REPRESENT THE IMPORTANCE OF FEATURES FROM DIFFERENT SOURCES. SVM IS FEASIBLE TO TUNE THOSE WEIGHTS BY TAKING THEM AS HYPER-PARAMETERS TO SELECT.EXPERIMENTS AND EVALUATIONS

A. Dataset Splitting

The CSIC dataset in which the duplicates have been removed as described in Section IV.A is split into training set, validation set and test set, where the sizes of three subsets are

described in Table IV. Among them, the training set accounts for 80%, the validation set for 10%, and the test set for 10%.

TABLE IV. DATA DISTRIBUTION

	Training	Validation	Test
Normal	7464	933	933
Anomalous	7138	892	893
Total	14602	1825	1826

B. Experiment Objectives

The experiment mainly includes the following two objectives:

- Evaluating the performance of various deep learning models used for malicious requests detection. To achieve this objective, deep learning models, such as TextCNN [20][7], LSTM [24], LSTM with attention [25][26] and RCNN [27], are built and compared using the same datasets.
- Verifying that SVM and transferable statistical features can improve the performance based on the high-dimension features extracted by TextCNN. Ablation experiments are made to demonstrate that both SVM and transferable statistical features are beneficial to improve detection results

To accurately evaluate the experimental results on test dataset described in Section VI.A, this paper utilizes the indicators of accuracy, precision, recall and F1-score to evaluate the results, as follows:

$$Accuracy = \frac{TP+TN}{TP+FP+\mathrm{FN}+TN} \tag{11}$$

$$Precision = \frac{TP}{TP+FP} \tag{12}$$

$$Recall = \frac{TP}{TP+FN} \tag{13}$$

$$F1-score = \frac{2*Precision*Recall}{Precision+Recall} \tag{14}$$

where TP (True Positive) is the number of anomalous requests detected, TN (True Negative) is the number of normal requests detected, FP (False Positive) is the number of misclassified normal requests and FN (False Negative) is the number of misclassified anomalous requests.

To obtain stable and fair experimental results, ten experiments of each model are performed to get an average result as the final output. Specifically, after the dataset is shuffled, two experiments are carried with the same parameters. The shuffling process is repeated five times, which leads to ten separate experiments. Subsequently, the average of the metrics in ten separate experiments are taken as the results in Table V and Table VI.

C. Result Analysis

1) Evaluating the performance of various deep learning models: Experiments are performed on different deep learning models. To ensure the fairness of the experimental results, all the deep models contain three common modules: (a) an embedding layer with 128 embedding dims; (b) a MLP that contains two fully connected layers in which the first fully connected layer with 128 dims followed by a ReLU activation function and the second fully connected layer with 2 dims followed by a Softmax function to transform the output to probability; and (c) using Adam optimization function, and cross-entropy as loss function. The deep learning models in the experiments are briefly described as follows:

- Bidirectional-LSTM: Gers at el. [24] create a model that concatenates the hidden outputs of each time-step t in bidirectional LSTM (hidden size of LSTM is 128, the same below) to obtain $[h_1, \dots, h_t, \dots, h_T]$ vector, and gets its average h_{ave}, then feeds h_{ave} into MLP to complete downstream classification tasks.
- LSTM-Attention [25]: Yang at el. create a model that has some differences from classical LSTM, which doesn't make average for concatenated hidden outputs h_t of each time-step, but makes weighted average instead as follows:

$$e_t = v^T \tanh(Wh_t + b) \tag{15}$$

$$a_t = \frac{\exp(e_t)}{\sum_i \exp(e_i)} \tag{16}$$

$$h_{weight} = \sum_{t=1}^{T} a_t h_t \tag{17}$$

  where the probability a_t, or its associated energy e_t reflects the importance of each hidden output h_t to the representation of the sentence and v, W, b are parameters learned during training.
- LSTM-Attention [26]: Xu at el. create a model which assumes that the last hidden output h_T contains enough information, so that the parameter v is not necessary and can be replaced by h_T, thus e_t can be calculated as follows:

$$e_t = h_T^T \tanh(Wh_t + b) \tag{18}$$

- RCNN: Lai at el. [27] create a model that is similar to TextCNN, but it uses LSTM to capture context information and concatenates LSTM's outputs with embedding before the convolution layer.

TABLE V. PERFORMANCE ON THE CSIC DATASET OF TRADITIONAL MACHINE LEARNING MODELS AND DEEP LEARNING MODELS

Method	EM	Naive Bayes	SOM	LSTM	LSTM+ Attention[25]	LSTM+ Attention[26]	RCNN	TextCNN
Accuracy	74.86%	84.08%	92.82%	97.17%	97.15%	96.92%	97.91%	98.48%
Precision	45.18%	66.96%	69.80%	97.56%	97.32%	96.79%	98.55%	99.62%
Recall	75.16%	52.35%	94.97%	96.67%	96.85%	96.94%	97.17%	97.26%
F1-score	0.5644	0.5876	0.8046	0.9711	0.9708	0.9686	0.9786	0.9843

- TextCNN: Kim at el. [20] propose the model as described in Section V.A. Referring to [20], in the experiment, the window size of convolution filter is set to {2, 3, 4, 5}， and the number of convolution filters of different widths is 64.

Fig. 7 shows the accuracies over 15 epochs when TextCNN model is trained. It is observed that after training 5 epochs, both the accuracies of training set and validation set have reached stable states. The TextCNN model with parameters at 12th epoch is saved due to the reason that the validation set gets the highest accuracy at 12th epoch. For TextCNN-SVM, TextCNN needs to have strong enough data representation capabilities and even can be over-fitting (i.e., focus on training accuracy) [28], then it is regularized using SVM. With this regard, the model with parameters at the last (15th) epoch is saved for the post-processing.

Table V shows both performances of deep learning models mentioned above and performances of tradition models—EM, Naive Bayes and SOM, cited from [29]. It is observed in Table VI that overall deep learning-based models (LSTM, LSTM + Attention and RCNN) achieve much better results than traditional machine learning-based models (EM, Naive Bayes, SOM) in terms of the four metrics, accuracy, precision, recall and F1-score, and TextCNN stands out of all models. More specifically, TextCNN reaches 98.48% on accuracy, 99.62% on precision, 97.26% on recall and 0.9843 on F1-score, which are the best performances in the experiments. Consequently, TextCNN has a good representational ability and it is in line with expectations for the reason that the malicious URL usually contains some anomalistic key-values pairs. TextCNN is adept at capturing locally important features to seize these anomalistic key-values pairs, while the model based on RNN is relatively better at capturing the long dependency of a sentence and the adversative relation in a sentence. Thus, TextCNN is more suitable for detecting malicious Web requests.

2) Experimental results of refinements based on TextCNN, compared with current methods: Table V shows the evaluations of refinements based on TextCNN and other approaches [8][9][10] in recent years.

- TextCNN-SVM improves accuracy by 0.45% (from 98.48% to 98.93%), and recall by 1.41%, compared with the baseline.
- TextCNN with transferable statistical features improves accuracy by 0.49%, and recall by 0.93%, compared with TextCNN.
- TextCNN-SVM with transferable statistical features achieves the best in terms of accuracy and recall, which improves accuracy by 0.85% (from 98.48% to 99.33%), and recall by 1.83%. Compared with other approaches, the proposed TextCNN-SVM with transferable statistical features achieves the state-of-art in terms of the evaluation indicators.

As shown in Table VI, Liang et al. [8] achieved 98.42% on accuracy, 97.56% on recall; Ito et al. [9] achieved 98.8% on accuracy; and Hao et al. [10] achieved 98.35% on accuracy, 99.0% on precision, 98.17% on recall. It is obvious that the proposed method in this paper outperforms those methods compared with above.

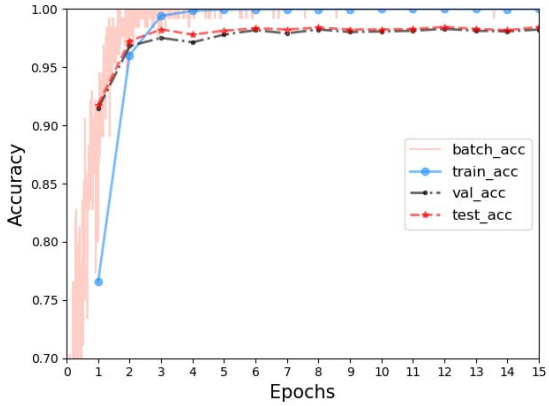

Fig. 7. TextCNN accuracy on batch/training/validation/test set during the training process

TABLE VI. EXPERIMENTAL RESULTS OF REFINEMENTS BASED ON TEXTCNN, COMPARED WITH CURRENT METHODS

Comparisons	Accuracy	Precision	Recall
TextCNN (Base)	98.48%	99.62%	97.26%
TextCNN -SVM	98.93%	99.15%	98.67%
TextCNN with Statistical Features	98.97%	**99.70%**	98.19%
TextCNN-SVM with Statistical Features	**99.33%**	99.53%	**99.09%**
Liang et al. [8]	98.42%	\	97.56%
Ito et al. [9]	98.8%	\	\
Hao et al. [10]	98.35%	99.00%	98.17%

VII. DISCUSSIONS

This section will further discuss the following two issues:

- How to visualize the abstract features extracted by TextCNN? This paper connects the features and words by extracting words-of-interest (WOI) to interpret the abstract features in Section VII.A.
- How to evaluate the impacts of data duplication? Section VII.B discusses the adverse effects of data duplication through experimenting.

A. Words-of-Interest Extracted by TextCNN

Although the features extracted by a deep learning model are abstract and difficult to explain, TextCNN is indeed able to capture the keywords through the output of max-pooling layer. The set of these keywords are defined as words-of-interest (WOI) in this paper. The keywords captured by the j-th convolution filter of TextCNN is defined as WOI^j, so WOI = $\{WOI^1, WOI^2, \dots, WOI^{|\boldsymbol{W}|}\}$, where $|\boldsymbol{W}|$ is the number of filters. WOI can explain to some extend what the neural network captures and what plays a relatively significant role in detecting malicious Web requests.

As described in Section V.A, the representation of a sentence is $\boldsymbol{x}[1:d]$ (Formula (2)), where d is the sentence's length. After a convolution operation of the j-th convolution filter with width s, the output is vector $\boldsymbol{m}^j = [m_1^j, \dots m_k^j, \dots m_{d-s+1}^j]$ (Formula (4)), in which m_k^l is corresponded to words $\boldsymbol{x}[k:k+s-1]$. With a max-pooling operation, one gets the feature value $\hat{m}^j$ (Formula (5)). WOI depends on the position k^j traced from $\hat{m}^j$, which is determined as follows:

$$k^j = \underset{k\in[1,d-s+1]}{\operatorname{argmax}} \{m_k^j | m_k^j \in \boldsymbol{m}^j\} \tag{19}$$

Based on k^j, the word positions in a sentence can be determined, i.e., WOI^j in the original URL turns out to be $\{word_{k^j}, word_{k^j+1}, \dots, word_{k^j+s-1}\}$.

For example, as shown in left side of Fig. 8, the sentence $\boldsymbol{x}[1:11]$ "*/ tienda1 / imagenes / nuestratierra . jpg / . inc*" has 11 words separated by spaces. A convolution operation involving filter **1** of width 2 is applied to the sentence to provide a feature vector $\boldsymbol{m}^1$. After a max-pooling operation, one gets the maximum value $\hat{m}^1 = \max\{\boldsymbol{m}^1\}$, whose position k^1 is 1. Correspondingly, the word positions in the sentence is [1, 2], i.e., $WOI^1 = \{/, tienda1\}$. Because there are three convolution filters, WOI includes $\{WOI^1, WOI^2, WOI^3\}$, specifically referring to $\{\{/, tienda1\}, \{/, .\}, \{., \text{inc}\}\}$ as shown in Fig. 8.

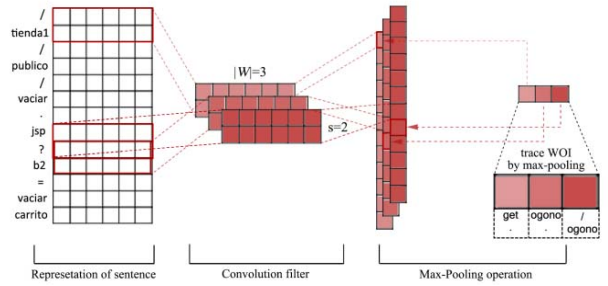

Fig. 8. Extracting the WOI by TextCNN

Let $count_k$ denote the number of times $word_k$ occurred in WOI, the weight of $word_k$ can be defined as:

$$weight_{word_k} = \frac{count_k}{max\{count_i | i\in[1,d]\}} \tag{20}$$

When $weight_{word}$ is greater than a threshold which is set to 0.3 in the experiment, the corresponding word is marked red as shown in Fig. 9.

(1) ...&username=<script>document.location='http://attackerhost.example/cgi-bin/cookiesteal.cgi?'+document.cookie</script>⋯
(2) ⋯/tienda1/publico/anadir.jsp?id=';waitfor delay'0:0:15';--&nombre=vinorioja&precio=100&cantidad=56⋯
(3) ...&dni=51834543h'; drop table usuarios; select * from datos where nombre like '%&...
(4) ⋯&nombre=galia&apellidos=sagradoespaña&email=hamissi@chica@sh@umed@&⋯

Fig. 9. Visualizing URLs by WOI

It is observed that WOI captured above such as "< script >", "document. location" indicate that URL (1) is an XSS attack to illegally obtain cookie information; URLs (2) and (3) belonging to SQL injection, "waitfor delay" is a SQL control command that can delay server responses, and "drop table" and "select * from…where…like…" show the malicious operations on the database. In addition, TextCNN can recognize an illegal pattern made up of ordinary words, such as "…@...@...@" in URL (4).

B. Impacts of Data Duplication

Generalization, one of the most important capabilities of a model, is defined as the performance on new, previously unseen data [30][31]. The CSIC dataset contains a large portion of repeated data, as shown in Table I, and the experiments in the previous sections are carried out using de-duplicated data. Some of previous researches performed experiments on the repeated data [7][9][10][11]. This section discusses what the impacts of data duplication using deep learning approaches are.

As shown in Table VII, the overlap rate is defined as the percentage of data in the test set which appear in the training set, and the overlap rate in the total test set reaches 92.27%.

TABLE VII. OVERLAP RATES OF DATA

	Training	Validation	Test	Overlap Rate
Normal	57600	7200	7200	95.11%
Anomalous	20052	2506	2507	84.12%
Total	77652	9706	9707	92.27%

As shown in Fig. 10, at first, the accuracy is calculated with the unseen data in the training set, and subsequently, data with the certain number of repetitions are gradually added to the test set, from 1, to 10, to 100 and finally to all of duplications in Table VII. It is observed that the accuracy with distinct data in the test set is 99.33%, however as the number of duplicated data in the training set increases, the accuracies are pushed up continuously, approaching 100%. The improvement on accuracy is not by optimizing the model itself, instead, by just duplicating data in the training set to the test set, which is a false-high performance.

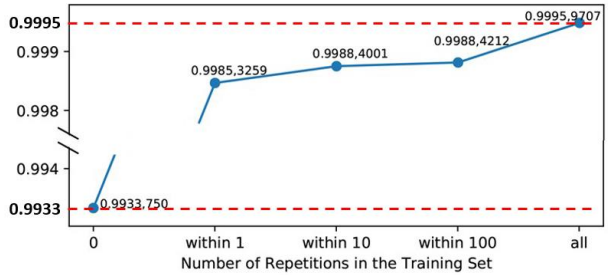

Fig. 10. Drawing up Accuracies with Repetition

VIII. CONCLUSIONS

This paper proposes an enhanced model by combining a deep learning-based method, TextCNN, and a traditional machine learning-based method, SVM, to detect malicious Web requests, where TextCNN is used for auto-extracting abstract features and SVM for classification at the last layer of TextCNN. To roll out the abstract features in terms of numeric vector, this paper defines trace-back functions, mapping max-pooling vectors back to the URL words, such that the words of interest to Web attack detection can be revealed instead abstract

numerical data in vectors. This facilitates researchers to understand and verify the results based on the machine learning approach. Furthermore, transferable statistical features are designed to boost representational power of SVM model. Both the abstract features extracted by TextCNN and transferable statistical features are fed into SVM for classification. To validate the method, several ablation experiments on HTTP Dataset CSIC 2010 are performed and the results show that the proposed approach outperforms the traditional machine approaches and deep learning approaches published in terms of accuracy, recall, precision and F1-score. In addition, as the CSIC dataset contains a large portion of duplication, the impacts of duplication in dataset are discussed, and reveal the false high accuracies that did happen in existing research.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their invaluable comments.

REFERENCES

[1] Verizon, 2015 Data Breach Investigations Report [Online], Available: http://www.iktissadevents.com/files/media/speeches/ACCF-2015-S4-lorenz-kuhlee.pdf, Accessed: 2020-01-14.

[2] Verizon, 2016 Data Breach Investigations Report [Online], Available: https://enterprise.verizon.com/resources/reports/DBIR_2016_Report.pdf, Accessed: 2020-01-14.

[3] A. D. Rayome, Report: Web application attacks up 69% in Q3 2017, here's what to do [Online], Available: https://www.techrepublic.com/article/report-web-application-attacks-up-69-in-q3-2017-heres-what-to-do/, Accessed: 2020-01-14.

[4] Vaadata, 2017 Statistics about Web-based Attacks, Mobile Attacks, Data Breaches [Online], Available:https://www.vaadata.com/blog/2017-statistics-web-based-attacks-mobile/

[5] C. T. Giménez, A. P Villegas and G. Á. Marañón, HTTP data set CSIC 2010, Information Security Institute of CSIC (Spanish Research National Council), 2010.

[6] A. Bochem, H. Zhang and D. Hogrefe, "Streamlined anomaly detection in web requests using recurrent neural networks," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2017, pp.1016-1017

[7] M. Zhang, B. Xu, S. Bai, S. Lu and Z. Lin, "A deep learning method to detect web attacks using a specially designed CNN," in *International Conference on Neural Information Processing*, Springer, Cham, 2017, pp.828-836.

[8] J. Liang, W. Zhao and W. Ye, "Anomaly-based web attack detection: a deep learning approach," in *Proceedings of the 2017 VI International Conference on Network, Communication and Computing*, ACM, 2017, pp.80-85.

[9] M. Ito, and H. Iyatomi, "Web application firewall using character-level convolutional neural network," in *2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA)*, IEEE, 2018, pp.103-106.

[10] S. Hao, J. Long and Y.Yang, "BL-IDS: Detecting Web Attacks Using Bi-LSTM Model Based on Deep Learning," in *International Conference on Security and Privacy in New Computing Environments*, Springer, Cham, 2019, pp.551-563.

[11] J. Wang, Z. Zhou and J. Chen, "Evaluating CNN and LSTM for Web Attack Detection," in *Proceedings of the 2018 10th International Conference on Machine Learning and Computing*. ACM, 2018, pp.283-287.

[12] A. M. Vartouni, S. S. Kashi and M. Teshnehlab, "An anomaly detection method to detect web attacks using Stacked Auto-Encoder," in *2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)*, IEEE, 2018, pp.131-134.

[13] H. T. Nguyen, C. Torrano-Gimenez, G. Alvarez, S. Petrović and K. Franke, "Application of the generic feature selection measure in detection of web attacks," in *Computational Intelligence in Security for Information Systems* Springer, Berlin, Heidelberg, 2011, pp.25-32.

[14] S. Pastrana, C. Torrano-Gimenez, H. T. Nguyen and A. Orfila, "Anomalous web payload detection: evaluating the resilience of 1-grams based classifiers," in *Intelligent Distributed Computing VIII*. Springer, Cham, 2015, pp.195-200.

[15] Z. Zhang, R. George and K. Shujaee, "Efficient detection of anomolous HTTP payloads in networks," in *SoutheastCon 2016*, IEEE, 2016, March, pp.1-3.

[16] N. Epp, R. Funk, C. Cappo and S. Lorenzo–Paraguay, "Anomaly-based web application firewall using HTTP-specific features and one-class SVM," in *Workshop Regional de Segurança da Informação e de Sistemas Computacionais*, 2017.

[17] G. Betarte, E. Gimenez, R. Martinez and A. Pardo, "Improving Web Application Firewalls through Anomaly Detection," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, 2018, December, pp.779-784.

[18] F. Mereani and J. M. Howe, "Exact and Approximate Rule Extraction from Neural Networks with Boolean Features," in *Proceedings of the 11th International Joint Conference on Computational Intelligence*, Scitepress, 2019.

[19] R. Kozik, M. Pawlicki, M. Choraś and W. Pedrycz, "Practical employment of granular computing to complex application layer cyberattack detection," in *Complexity*, 2019.

[20] Y. Kim, "Convolutional Neural Networks for Sentence Classification," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp.1746-1751

[21] C. Cortes and V. Vapnik, "Support-vector networks," in *Machine learning*, 1995, 20(3), pp.273-297.

[22] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of Workshop at the International Conference on Learning Representations (ICLR),* 2013.

[23] L. V. D. Maaten and G. Hinton "Visualizing data using t-SNE," in *Journal of machine learning research*, 2008, pp.2579-2605.

[24] F. A. Gers, J.Schmidhuber and F. Cummins, "Learning to forget: continual prediction with lstm," *Neural Computation*, 2000, 12(10), pp.2451-2471.

[25] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola and E. Hovy, "Hierarchical attention networks for document classification," in Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies, 2016, pp.1480-1489.

[26] J. Xu, X. Sun, Q. Zeng, X. Ren, X. Zhang, H. Wang and W. Li, "Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach," 2018, arXiv preprint arXiv:1805.05181.

[27] S. Lai, L. Xu, K. Liu and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[28] A. Karpathy, A Recipe for Training Neural Networks [Online], Available: http://karpathy.github.io/2019/04/25/recipe/, Accessed: 2020-01-14.

[29] Le Jr, D. 2017. An unsupervised learning approach for network and system analysis. Dalhousie University.

[30] G. Ian, B. Yoshua, and C. Aaron, *Deep Learning.* MIT Press, 2016.

[31] Z.-H Zhou, *Machine Learning.* Tsinghua University Press, 2016.

[32] M. V. Mahoney and P. K. Chan, "An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection," in *International Workshop on Recent Advances in Intrusion Detection,* 2003, Springer, Berlin, Heidelberg, pp.220-237.

[33] Analyzing Web Traffic ECML/PKDD 2007 Discovery Challenge [Online], Available: http://www.lirmm.fr/pkdd2007-challenge/index.html, Accessed: 2020-01-14.

[34] Network Working Group, RFC (Request For Comments) -1738: Uniform Resource Locators (URL). [Online], Available: http://www.faqs.org/rfcs/rfc1738.html, Accessed: 2020-01-14.

[35] X. Glorot, A. Bordes and Y. Bengio, "Deep sparse rectifier neural networks," in Proceedings of the *fourteenth international conference on artificial intelligence and statistics*, 2011, pp.315-323.