Evaluating CNN and LSTM for Web Attack Detection

Jiabao Wang, Zhenji Zhou, Jun Chen
Army Engineering University
Guanghua Road, Haifu Streat, No.1
Nanjing, China 210007
jiabao_1108@163.com,zhou_zhenji@163.com,cjcqsports@163.com

ABSTRACT
Web attack detection is the key task for network security. To tackle this hard problem, this paper explores the deep learning methods, and evaluates convolutional neural network, long-short term memory and their combination method. By comparing with the traditional methods, experimental results show that deep learning methods can greatly outperform the traditional methods. Besides, we also analyze the different factors influencing the performance. This work can be a good guidance for researcher to design network architecture and parameters for detecting web attack in practice.

CCS Concepts
• Security and privacy→Web protocol security • Computing methodologies→Neural networks.

Keywords
Network security; web attack detection; convolutional neural network; recurrent neural network; long-short term memory.

1. INTRODUCTION
Web attack detection is one of the most important cyber-security problems. In web communication, the malicious behavior is hard to detect, because it is very similar to the normal behavior [1]. Fortunately, most of the web attack behaviors are based on the various known web protocols. And the widely used protocol for web attack is based on the HTTP protocol. So in this paper, we focus on the HTTP communication to predict web attack.

How do we distinguish the malicious web communications from the normal ones? In the past, detection of web attack is based on matching key words or key patterns, which are always determined by the security domain experts [2]. This method is very effective and accurate to find web attack. But it depends on the expert's experience, and the definition of key words and key patterns is very hard. Besides, in practice, there are many (hidden) patterns which are hard to find by the domain experts, especially on the large-scale dataset. As a result, how to find web attack automatically by computer is the key problem.

ICMLC 2018, February 26–28, 2018, Macau, China

ACM ISBN 978-1-4503-6353-2/18/02…$15.00
DOI: https://doi.org/10.1145/3195106.3195107

Recently, deep learning technology has been achieved great progress in feature representation and classification, due to the collections of big data and the strong GPU computing resources. As we known, in deep learning, Convolutional Neural Network (CNN) [3] and Recurrent Neural Network (RNN) [4] are the two famous and popular methods to model the spatial and temporal data. They have the end-to-end architecture without designing features manually. So it is a straight idea to use deep learning methods for solving the difficult web attack detection problem on the large-scale dataset. In this paper, we explore these two methods and their combination for detecting web attack.

CNN [3,5] is mainly composed of convolution and pooling operations. Convolution operation can find the spatial pattern and relationship. It can produce a lot of space invariant discriminative features with low computation based on the shared-weights architecture. Pooling is a sub-sampling operation, which can extract the important value and further decrease the computation in the subsequent operations. A cascaded sequence of convolution and pooling operations can build an end-to-end and powerful deep neural network.

RNN [4,6] is a method for modeling sequence or temporal data. It can find the long-term relationship based on the memory unit. In the various existing RNNs, Long-Short Term Memory (LSTM) [7,8] is the most famous one and achieves great success in text and speech translation. LSTM consists of input gate, output gate, forget gate and candidate memory cell. It can avoid the long-term dependence problem with the special designed memory unit.

In text or sentence classification, CNN and LSTM have achieved great success [3,8]. However, in web security, few researchers have explored CNN and LSTM to handle the detection of web attack, such as [9,10]. It is mainly because the web request string is a concatenating string without space separator and cannot be treated as text directly. So in this paper, we explore web attack detection by adopting CNN and LSTM. We proposed the basic CNN and LSTM network architectures and also proposed the combination architecture. Experiments show that the proposed models can outperform the traditional classification methods greatly on the public CSIC2010 dataset.

2. DATA PRESENTATION
As we known, the request of web communication can be treated as a command string, so we can detect web attack by classifying the request string. Due to the various methods for text classification, we try to transform the string into text, which is composed by a sequence of words. Here, based on the prior knowledge of web request, we can separate the string by the rule of the protocols. For example, HTTP protocol has many segments separated by symbol '\&', and the keys and values are separated by '=' in each segment. As a result, we can separate the request string into a sequence of 'words' or symbols, even the 'words' have no meanings literally. Deep learning methods, CNN and

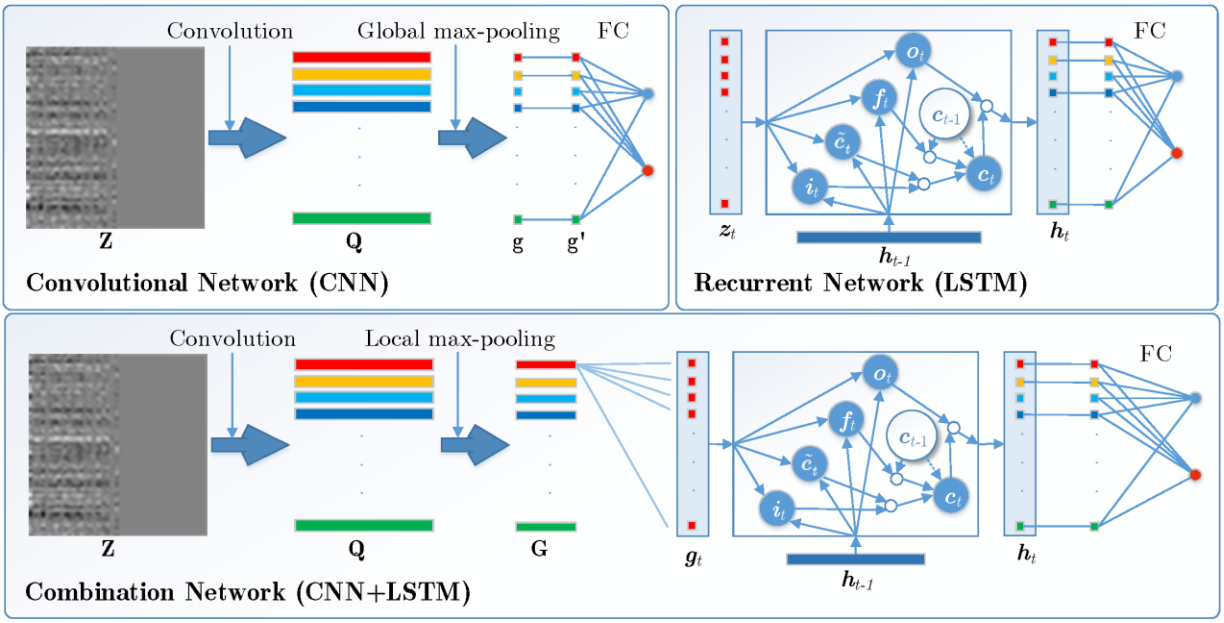

Figure 1. Architecture of the three networks

RNN, can find the relationship on these 'words', and build a model to classify the attack and normal web communication requests.

After 'words' segmentation, we can represent each split 'word' as a one-hot vector, liking the representation in document classification. The one-hot vector is a binary vector, which the i th element for the i-th 'word' in the vocabulary is set to one and all others are set to zeros. As a result, the sequence with L 'words' is represented as a sequence of L one-hot vectors $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_L)$. In reality, the one-hot vector for each word is a binary, sparse and high-dimensional vector. It is time-consuming for computation and is difficult to measure the relationship between two 'words', which may have synonymous in semantic space.

Inspired by the word embedding [11,12], we project the one-hot vector $\mathbf{x}_i$ into a low d-dimensional continuous vector $\mathbf{z}_i \in R^d$. It can be done just by multiplying the vector $\mathbf{x}_i$ from left with a weight matrix, $\mathbf{M} \in R^{d \times |V|}$, where $|V|$ is the number of unique words in the vocabulary:

$$\mathbf{z}_i = \mathbf{M}\mathbf{x}_i. \tag{1}$$

This matrix $\mathbf{M}$ for word embedding can be obtained by random assignment or by learning a network with one-hidden layer. Here, we train a network by inputting a word (one-hot vector) and outputting the next word (one-hot vector) to learn the relation of the two co-occurrence words. Then the low-dimensional, real-valued hidden outputs in the trained network can be used to present the one-hot word vector. In experiment, we find that the learned low-dimensional, real-valued representation achieves great performance. Once done the word embedding, the input sequence of one-hot vectors becomes a sequence of dense, real-valued vectors, which can be noted as a matrix $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_L)$.

3. EVALUAITON METHODS

3.1 Convolutional Network

For the sequence $\mathbf{Z}$, a set of n filters $\mathbf{F} = \{\mathbf{f}_j \in R^{d \times h}\}_{j=1}^{n}$ with receptive field size h, is applied in the convolutional network. For each filter $\mathbf{f}_j$, the result $\mathbf{q}^j$ of the t position as

$$q_t^j = \phi(\mathbf{f}_j^T \mathbf{z}_{t:t+h-1} + b), \tag{2}$$

where $\mathbf{q}^j \in R^{1 \times L-h+1}$, b is the bias and ϕ is the nonlinear rectify function. For all n filters in $\mathbf{F}$, there is total n sequences in $\mathbf{Q} = \{\mathbf{q}^j\}_{j=1}^{n}$. As a result, the convolutional result $\mathbf{Q}$ is a new sequence with width n and length $L-h+1$.

After convolution, pooling is applied to the sequence. A global max-pooling operation $g^j = max_{t \in \{1,...,L-h+1\}}\{q_t^j\}$ is always used in practice. The computation is applied on the length dimension, so the result of max-pooling is a sequence of size n. This operation can capture the most important characteristic (the highest value) in the sequence for each feature map produced by each filter. And more, this operation can deal with the variable sequence lengths. The result of convolutional and pooling operation is a vector $\mathbf{g} = (g^1, g^2, ..., g^n)$, which can be mapped to two outputs for giving the probabilities of web attack and normal communication.

3.2 Recurrent Network

The recursive function ψ in the recurrent network takes one vector $\mathbf{z}_t$ and the previous hidden state $\mathbf{h}_{t-1}$ as the input, and returns the new state:

$$\mathbf{h}_t = \psi(\mathbf{z}_t, \mathbf{h}_{t-1}), \tag{3}$$

where $\mathbf{z}_t \in R^d$ is the vector of input sequence $\mathbf{Z}$ at time t. The start hidden state $\mathbf{h}_0 \in R^n$ is often initialized as an all-zero vector.

In practice, the most naive recursive function is implemented as

$$\mathbf{h}_t = tanh(\mathbf{W}_z \mathbf{z}_t + \mathbf{U}_h \mathbf{h}_{t-1}), \quad (4)$$

where $\mathbf{W}_z \in R^{n \times d}$ and $\mathbf{U}_h \in R^{n \times n}$ are the weight matrices. However, this function suffers from the problem of vanishing gradient.

Nowadays, it is common to use LSTM unit to learn the long-term dependencies in the flow information to prevent the vanishing gradient. LSTM unit consists of four sub-units: input gate, output gate, forget gate and candidate memory cell. They are computed by

$$\begin{aligned} \mathbf{i}_t &= \sigma(\mathbf{W}_i \mathbf{z}_t + \mathbf{U}_i \mathbf{h}_{t-1}) \\ \mathbf{o}_t &= \sigma(\mathbf{W}_o \mathbf{z}_t + \mathbf{U}_o \mathbf{h}_{t-1}) \\ \mathbf{f}_t &= \sigma(\mathbf{W}_f \mathbf{z}_t + \mathbf{U}_f \mathbf{h}_{t-1}) \\ \text{¢}_t &= tanh(\mathbf{W}_c \mathbf{z}_t + \mathbf{U}_c \mathbf{h}_{t-1}) \end{aligned} \quad (5)$$

The output or activation of the LSTM unit is computed as

$$\mathbf{h}_t = \mathbf{o}_t \text{ e } tanh(\mathbf{c}_t), \quad (6)$$

where $\mathbf{c}_t = \mathbf{i}_t \text{ e } \text{¢}_t + \mathbf{f}_t \text{ e } \mathbf{c}_{t-1}$. The output sequence of the recurrent network is $(\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_L)$. Usually, the last output vector $\mathbf{h}_L$ is always directly mapped to two outputs for binary classification.

3.3 Combination Network

As we known, CNN can extract the abstract, local translation-invariant higher-level features by its convolution and pooling operations, and LSTM can capture the long-term dependencies by its special designed memory unit. It's a simple idea to combine both wonderful methods.

According to Section 3.1, the result produced by the convolution and pooling is a vector, which cannot be directly combined with LSTM. To tackle this problem, we replace the global max-pooling with a local max-pooling, which also produce a sequence for n filters. The local max-pooling result at t position is formulated as

$$g_t^j = max_{t \in \{t, ..., t+h'-1\}} \{q_t^j\}, \quad (7)$$

where max is applied to each $\mathbf{q}^j$. Finally, it produces a new sequence of vectors as

$$\mathbf{G} = (\mathbf{g}_1, \mathbf{g}_2, ..., \mathbf{g}_{L'}), \quad (8)$$

where $L' = (L - h + 1) / h'$ and $\mathbf{g}_t = (g_t^1, g_t^2, ..., g_t^n)^T$.

After this, the sequence of feature vectors $\mathbf{g}_t$ is fed into LSTM to finish the combination classification.

The CNN, RNN and their combination can map the sequence of 'word' vectors into a new vector, which is usually fully connected to two outputs. For binary classification, the larger one in the two values corresponds to the predicted class. The softmax log-loss is adopted as the optimization objective

$$L = \frac{1}{N} \sum_{i=1}^{N} \left(-\log \frac{exp(y_i^{t_i})}{\sum_{k=1}^{C} exp(y_i^k)} \right), \quad (9)$$

where $\mathbf{y} = \{y_i\}_{i=1}^n$ is the input of softmax log-loss function, $y_i^{t_i}$ is the value of y_i at the t_i position. N is the number of samples.

3.4 Implementation

The architecture of CNN, LSTM and their combination are presented in Fig. 1. The input of these three methods is the 'word' embedding. Each word has the dimension of 40. The sequence of the request is fixed to 56 'words'.

For CNN, there is a convolutional layer with 100 filters of width 5, and a global max-pooling layer. Then, a dropout layer with rate 0.9 is used to suppress over-fitting, and finally a fully connection layer is adopted to map into a binary values for classifying attack and normal web request. For LSTM, the hidden variable has the dimension of 100, and the last output is connected to a dropout layer with rate 0.9. The two outputs are achieved by a fully-connection layer. For the combination (CNN+LSTM), the local max-pooling layer replaces the global max-pooling layer. The local max-pooling size is 5 and the output is a sequence which is transferred to the next LSTM network.

4. EXPERIMENTS

4.1 Dataset

The experiments are conducted on CSIC2010 dataset[1], which is developed by the Information Security Institute of CSIC (Spanish Research national Council). The dataset contains tens of thousands of HTTP protocol requests, which targeted to an e-Commerce web application. The published data was split into training (only normal) and testing (anomalous and normal) sets. There are over 36000 normal and 25000 anomalous requests in the testing set, which is used in our experiments. The anomalous requests contains a large range of web attacks, such as SQL injection, buffer overflow, information gathering, files disclosure, CRLF injection, XSS and parameter tampering. The requests targeting hidden (or unavailable) resources are also considered as anomalies [13,14].

Due to our knowledge, there is no other publicly available dataset for the web attack detection problem. The datasets like DARPA or KDD'99 are outdated and do not cover many of the current attacks.

4.2 Results

We compared our algorithm with the traditional methods and the deep learning methods. The traditional methods include Multinomial Naive Bayes (NB), Linear Support Vector Machine (Linear SVM), and Neural Network (NN), k-Nearest Neighbour (kNN), Decision Tree (DT). The deep learning methods include CNN, LSTM and their combination (CNN+LSTM). All of the traditional methods adopt the TF-IDF vector features, while the deep learning methods use the embedding vector, which is described in Section 2.

The parameters of the traditional methods are set as follows:

For NB, the constant parameter alpha is set as 0.01. For Linear SVM, the linear classifier is adopted. For NN, we used two hidden

[1] http://www.isi.csic.es/dataset/

layers with 50 and 10 hidden units. For kNN, the neighbor parameter is set as 3. DT, the entropy is used and the minimal number of samples in a leaf is set as 3.

To training the deep learning methods, we set the batch size to 128, and the learning rate is set to 0.001. The epochs of training as 10, and the dropout rate is set to 0.9. The 'adam' optimizer is used for learning.

These methods are evaluated by the 10-fold cross-validation technique. The dataset is divided randomly into 10 sets. One set is used for evaluation while the remaining is used for training. The results for 10 runs (10-folds) are averaged to produce the overall performance.

The results are reported in Table 1. From this table, we can find that deep learning methods outperform all traditional methods greatly. Furthermore, the combination method (CNN+LSTM) achieves the best performance.

Table 1. Comparison with state-of-the-art methods

	Precision	Recall	F1-Score
CNN+LSTM	**0.989**	**0.988**	**0.989**
CNN	0.986	0.983	0.985
LSTM	0.977	0.979	0.978
DT	0.925	0.917	0.920
kNN	0.907	0.911	0.908
NN	0.895	0.882	0.887
Linear SVM	0.887	0.867	0.875
NB	0.767	0.776	0.765

4.3 Analysis

In this subsection, we analyze the performance of deep learning methods from the following aspects:

4.3.1 Convolutional Network Analysis

As we known, the number and the width of the filters in the convolutional network are the key parameters. We test different number of filters and the width of filters. Firstly, we set the number of filters as 20, 50, 100, 200 and 300 for testing. The results are presented in Fig. 2(top). Secondly, we set the width of filters as 3, 5, 7, 9 and 11. The results also presented in Fig. 2(bottom).

From the Figure, we can find that the performance is increase as

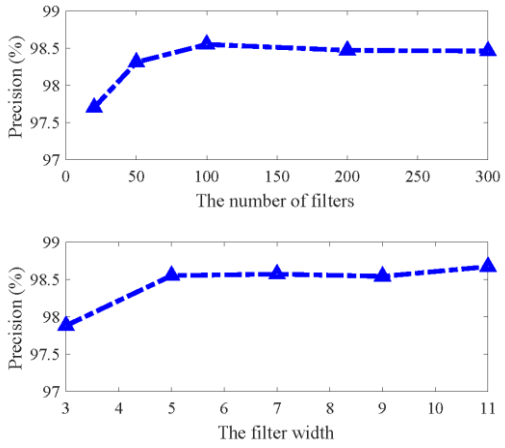

Figure 2. Performance about CNN with (top) different filter number and (bottom) different filter width

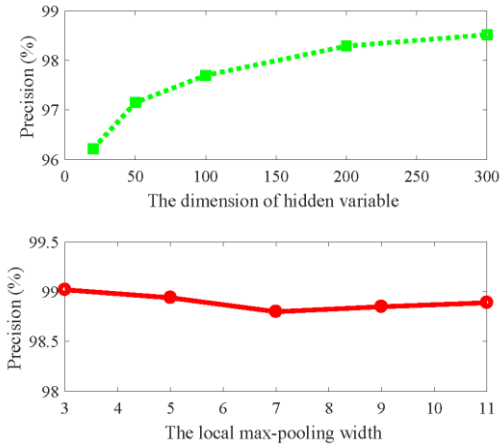

Figure 3. Performance about (top) LSTM method with different dimension of hidden variable and (bottom) Combinaiton method with different local max-pooling width

the number of filters increase. The best performance is achieved with 100 filters. The performance increase slowly when the width of filter is larger than 5.

4.3.2 Recurrent Network Analysis

For LSTM, the key parameter is the dimension of the hidden variable. In this part, we test different dimensions to evaluate the performance of LSTM method. The results are presented in Fig. 3(top). From the figure, we can conclude that LSTM can achieve better performance when the dimension increases in the range [20, 300]. However, the time-consuming also increases as the dimension increases. It is better to choose a balance between the performance and the computation.

4.3.3 Combination Network Analysis

In this part, we test the performance of the combination method (CNN+LSTM) with different local max-pooling size. Fig. 3(bottom) gives the results. From the figure, we can also find that the smaller local max-pooling size in CNN+LSTM can achieve better performance. The best performance is achieved when the size is set to 3.

4.3.4 Dropout Strategy Analysis

Considering the over-fitting problem on the small-scale CSIC2010 dataset, we analyze influence of the dropout strategy. Fig. 4 gives the results of different dropout rate with different

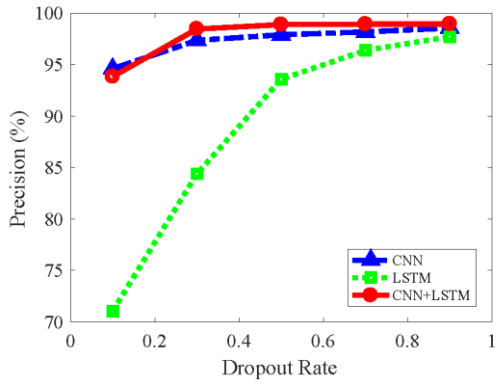

Figure 4. Performance of different methods with different dropout rate

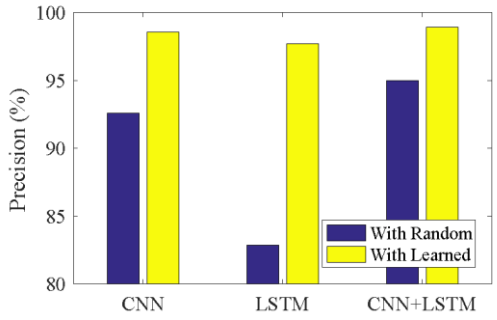

Figure 5. The performance of different word embeddings with random initialization (With Random) and the learning method (With Learned)

methods. From the figure, we can find the dropout strategy has great ability to suppress the over-fitting and achieves better performance. And a larger rate corresponds to a better performance. Among the three methods, LSTM has a large variation with different dropout rate. While CNN and CNN+LSTM methods are robust to different dropout rate. Besides, the performance increases slowly when the rate is larger than 0.3.

4.3.5 Word Embedding Analysis

As the representation of the word embedding is also a key factor. We test two different word embedding techniques: random assignment or by learning a network with one-hidden layer. The comparison is presented in Fig. 5. From the figure, we can find that the method with learning can achieve better performance.

5. CONCLUSION

In this paper, we introduce deep learning methods, CNN and LSTM, to detect web attack. We compared them with several traditional methods. Experimental results show that deep learning methods can greatly outperform the traditional methods. Besides, we also analyze the different factors influencing the performance. This work can be a good guidance for researcher to design network for detecting web attack.

6. ACKNOWLEDGMENTS

This work is supported by the Chinese national key research and development plan fund project. (2017YFB0802900).

7. REFERENCES

[1] J. Kim, D. H. Yoo, H. Jang, and K. Jeong, Webshark 1.0: a benchmark collection for malicious web shell detection. *Journal of Information Processing Systems*, 11(2):229-238, 2015.

[2] O. Starov, J. Dahse, S. S. Ahmad, T. Holz, and N. Nikiforakis, No honor among thieves: a large-scale analysis of malicious web shells. In *Proceedings of the International Conference on World Wide Web*, pages 1021-1032, 2016.

[3] Y. Kim, Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[4] T. Mikolov, M. Karafit, L. Burget, J. Cernocky, and S. Khudanpur, Recurrent neural network based language model. In *Proceedings of the Annual Conference of the International Speech Communication Association*, Makuhari, Chiba, Japan, September, pages 1045-1048, 2010.

[5] C. N. D. Santos and M. Gattit, Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of the International Conference on Computational Linguistics*, 2014.

[6] P. Liu, X. Qiu, and X. Huang, Recurrent neural network for text classification with multi-task learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2873-2879, 2016.

[7] H. Sak, A. Senior, and F. Beaufays, Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *Computer Science*, pages 338-342, 2014.

[8] J. Nowak, A. Taspinar, and R. Scherer, Lstm recurrent neural networks for short text and sentiment classification. In *Proceedings of the International Conference on Artificial Intelligence and Soft Computing*, pages 553-562, 2017.

[9] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, Long short term memory recurrent neural network classifier for intrusion detection. In *Proceedings of the International Conference on Platform Technology and Service*, pages 1-5, 2016.

[10] Y. Tian, J. Wang, Z. Zhou, and S. Zhou, Cnnwebshell: Malicious web shell detection with convolutional neural network. In *Proceedings of the International Conference on Network Security*, 2017.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[12] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, Distributed representations of words and phrases and their compositionality. In *Proceedings of the Advances in Neural Information Processing Systems*, 26:3111-3119, 2013.

[13] C. Torrano-Gimenez, A. Perez-Villegas, and G. Alvarez, An anomaly-based approach for intrusion detection in web traffic. *Journal of Information Assurance and Security*, 5(4):446-454, 2010.

[14] T. N. Hai, C. Torrano-Gimenez, G. Alvarez, S. Petrovic, and K. Franke, Application of the Generic Feature Selection Measure in Detection of Web Attacks. *Springer Berlin Heidelberg*, 2011.