IET Information Security

Research Article

Leveraging deep neural networks for anomaly-based web application firewall

ISSN 1751-8709
Received on 31st July 2018
Revised 23rd October 2018
Accepted on 20th November 2018
E-First on 28th February 2019
doi: 10.1049/iet-ifs.2018.5404
www.ietdl.org

Ali Moradi Vartouni[1], Mohammad Teshnehlab[1] ✉, Saeed Sedighian Kashi[1]
[1]Faculty of Computer Engineering, K.N. Toosi University of Technology, Seyed Khandan, Shariati Ave, Tehran, Iran
✉ *E-mail: teshnehlab@eetd.kntu.ac.ir*

Abstract: Web applications are the most common platforms for the exchange of information and services on the Internet. With the launch of web 2.0, information has flourished through social networking and business online. Therefore, websites are often attacked directly. As a result, the industry has paid more attention to the security of web applications in addition to security under computer networks. Intelligent systems based on machine learning have demonstrated excellent result on tasks such as anomaly detection in web requests. However, current methods based on traditional models cannot extract high-level features from huge data. In this study, the authors proposed methods based on deep-neural-network and parallel-feature-fusion that features engineering as an integral part of them and plays the most important role in their performance. The proposed methods use stacked autoencoder and deep belief network as feature learning methods, in which only normal data is used in the classification of training phase, then, one-class SVM, isolation forest, and elliptic envelope are applied as classifiers. The authors compared the proposed model with different strategies on CSIC 2010 and ECML/PKDD 2007 datasets. Results show that deep model and feature fusion model demonstrated as hierarchical feature learning which had better performance in terms of accuracy and generalisation in a reasonable time.

1 Introduction

Protection of computers and networks from theft, infiltration, and disruption is one of the key problems in different domains of computer science. The significance of computer system security grows with the increment of internet users. Many efforts such as intrusion detection system (IDS) and firewall have been made to establish various security solutions. Firewalls and IDSs in the network layer usually do not examine communication packets in the application layer. Hence, they are not adequately capable of protecting the web servers. Web applications are one of the most attractive targets for attackers to infiltrate the information infrastructure of organisations. Failure to provide web security for organisations has implications for financial and credit losses, internal data leaks, and website manipulation. A web application firewall (WAF) is a tool to identify and prevent many types of attacks, such as XSS and SQL injection. It is used on web applications to prevent information leakage of organisations. WAFs use IDS methods in the application layer to secure web applications.

The survey in [1] gives a more elaborate image and comprehensive review of IDSs. In general, the IDS methods are divided into two groups based on signature and anomaly detection. The signature-based approach tries to find a pre-identified pattern in packet content. This method is usually simple and effective for known attacks, but the method is an inefficient facing with new attacks while updating signatures. The second type of IDSs, anomaly detection approach looks for a pattern that does not conform the expected behaviour using machine learning and pattern recognition methods. These algorithms have the advantage that they can detect new types of intrusions as deviations from normal behaviour. The survey in [2] has reviewed many anomaly detection techniques which had been specifically developed for certain application domains.

One of the key problems of IDSs based on machine learning is the large volume of data and time complexity of algorithms. Many IDSs perform a feature engineering step to extract a subset of relevant features from the traffic dataset to enhance detection results. Effectiveness and performance of IDS are highly dependent on the quality of the feature construction and feature selection algorithms. Association rule learning, frequency episode extraction, and *n*-grams extraction are three commonly used methods for feature construction phase for IDSs [3]. The number of features increases exponentially with *n* of *n*-gram model. Thus, it is necessary to consider solutions to deal with the curse of dimensionality and computational complexity of the problem. PayL [4] analyses packet payload to model byte frequency distribution which models 1-gram distributions for normal traffic. Anagram [5] models a mixture of high order *n*-grams ($n > 1$) and stores portions of legitimate traffic in the form of *n*-grams into efficient hash maps. McPAD [6] deals with high order *n*-gram which designed an ensemble of multiple one-class SVM (1SVM) classifiers to reduce the dimensionality of the feature space, which is applied as a feature clustering algorithm.

Deep learning is particularly good at a type of learning that has the potential to be very useful for real-world applications such as speech recognition, computer vision, and natural language processing. A good data representation has high performance for a machine learner, and deep learning algorithms extract automated complex data representation at high levels of abstraction. Researchers have applied deep learning for the implementation of various IDSs. SCDNN [7] combines spectral clustering and deep neural network (DNN) algorithms for intrusion detection in sensor networks. Also, a distributed denial of service (DDoS) detection system that incorporates stacked autoencoder (SAE) in a software-defined networking environment has been implemented in [8]. They use deep learning for feature reduction of a large set of features derived from network traffic headers.

Detection of web-based attacks has recently received considerable attention because of the increasingly critical role that web application and services are playing. In this paper, we propose a WAF based on anomaly detection using machine learning algorithms. Our approach analyses the HTTP traffic in three phases: feature construction, feature extraction, and detection of anomalies, respectively. In the first phase, to construct features from HTTP data, the character-based *n*-gram model is applied. To reduce the dimensionality of the problem, DNNs are implemented to extract relevant features from data. Also, to enrich feature sets of datasets, DNN and feature fusion are considered. Finally, a classifier is applied to detect anomalies.

To the best of our knowledge, this paper is a proper study in which DNN is applied to the WAF and shows better performance regarding accuracy, generalisation, and speed compared with other considered models.

The contributions of this paper are as follows:

- We apply three one-class anomaly detection methods including 1SVM, ensemble isolation forest (IF), and covariance elliptic envelope (elliptic) under two HTTP datasets.
- We apply feature extraction methods including principal component analysis (PCA), kernel PCA (KPCA), fast independent component analysis (ICA) (FICA), and DNN models namely SAE, and deep belief network (DBN) to features obtained from 2-gram representation.
- Finally, we use a fusion of features which are extracted from 1-gram and 2-gram in parallel for detection model.

This paper is organised as follows. After the introduction, Section 2 includes related works in the WAFs based on anomaly detection. Section 3 considers the architecture of the proposed system. We describe our experiments and results in Section 4 and present discussions in Section 5 followed by concluding remarks in Section 6.

2 Related works

After that Kruegel and Vigna [9] introduced a web attack detection system based on HTTP traffic, due to its importance in network security, other researchers began to investigate this issue. In this section, a brief review about anomalies detection in the context of web traffic, HTTP, security is presented. The work presented here is related to three different areas of WAF, namely feature construction from HTTP data, feature extraction, and machine learning-based anomaly detection.

To construct feature from the web request, Kruegel and Vigna [9] have analysed statistical characteristics of HTTP traffic. They modelled these characteristics by selecting six groups of parameters from requests. These models concentrate on individual queries of requests and their parameters. After that, Kruegel *et al.* [10] have extended the work of Kruegel and Vigna [9] and introduced three additional statistics which have focused on patterns of whole sequences of queries. Another feature construction method, so-called tokenisation, has been introduced by the authors in [11, 12]. They tokenised HTTP request and employed deterministic finite automata to detect malicious web requests. *N*-gram is also used by spectrogram method in [13] to characterise the distribution of content and structure of HTTP. They applied a mixture of multiple Markov chains to recognised legitimate web layer script inputs. Spectrogram reduces the exponentially growing sample space into a linearly growing space. In [14], two types of statistics: per request and per time bin are extracted from HTTP logs; they used *n*-gram to extract following parameters from the request: resource request, values of the query's attributes, and user agent.

Structure and feature extraction are key points of WAFs. GSAD model [15] analysed the correlation between various 1-gram features and used Mahalanobis distance map to calculate the difference between normal and abnormal network traffic. An algorithm for filter-based feature selection is also provided in [16–18]. In their work after extracting *n*-gram or expert knowledge from dataset, correlation feature selection measure and minimal redundancy-maximal relevance measure are applied to filter the data. An effective method proposed by Wang and Zhang [19] extracts exemplars from HTTP traffic before the detection model is built. The affinity propagation is employed for dynamic clustering to extract the exemplars from HTTP traffic without the need to specify the number of clusters. In the following, they employed the affinity propagation algorithm to fulfil autonomic intrusion detection that detects intrusions in an online and adaptive fashion through dynamic clustering of data streams [20]. An approach based on signal processing techniques, considering wavelet methods, is proposed by the authors in [21, 22].

Neural network and fuzzy-set approaches have been widely used in several works such as [23] for web applications firewalls and for anomaly detection in special-purpose attack such as DDoS [24, 25]. A neuro-fuzzy classifier is designed to discriminate classes in [26] and to limit the statistical parameters of each fuzzy rule which uses self-organising map clustering. The extreme learning machine is proposed in [27] in order to distinguish normal and abnormal web traffic.

Due to the importance of inferential methods for minimising false positives, Markov's methods have been used in many studies to detect web attacks [28]. Two Markov models are introduced in [29]; one for detection of injection attacks and the other for dealing with legitimate web service client behaviour. An online learning statistical model is proposed in [30], they transformed web request into an *n*-tuple of symbols. Prediction by partial modelling is applied to determine if the next symbol is dependent on the previous symbol. Then, by traveling the built tree with a depth of *D*, all *n*-grams are found from 1-gram to *D*-gram. In addition, probability-distributed model, hidden Markov model, and 1SVM model are introduced in [31] for WAF.

Ontological techniques are approaches that provide semantic learning. An ontology system can represent a graphical representation of the model using concepts and instances, properties, relationships, rules, and model interpretations. Two ontology-based approaches are proposed in [32, 33] to detect and classify attacks and communication protocols for web applications. This method can remove inconsistency, incompleteness, and redundancy in the specification of ontological concepts.

Researchers have applied deep learning based on SAE for the implementation of WAF [34], denoising SAE [35], recurrent neural network including long-short-term-memory and gated-recurrent-unit [36], and character-level convolutional neural network for web attacks detection [37].

The HTTP data is non-stationary; thus, statistical analysis of HTTP and tokenised HTTP request does not have a comprehensive approach to data description. On the other hand, *n*-gram method grows as *n* grows exponentially. As a result, we need to introduce a feature engineering for modelling. Many of the works are used in statistical analysis or tokenised or even 1-gram for construction of features. We use 2-gram method and apply DNN models to extract relevant features. DNNs are trying to model the high-level abstractions in the data. These models are useful for developing data representation.

3 Methodology

The proposed framework works through unsupervised algorithms over HTTP request data. The proposed method builds a model for only normal data as in one-class classifiers such as 1SVM because we have to build our model without using abnormal data to robust the model against new unknown attacks. Due to skewed data and the ability of prediction of future unknown attacks, it had better use one-class anomaly detection methods. The main steps of the method are illustrated in Fig. 1.

3.1 Feature construction

Once user requests have been received, they are sent to the WAF server as part of an HTTP request. To detect attacks against a web server, it is necessary to construct features from HTTP logs which contain method, URI, query, HTTP version, headers, cookies, and body. To extract features from HTTP request, *n*-gram character-based model is applied. The *n*-gram model, like many statistical models, depends on the training data. A character-based *n*-gram is a subsequence of *n* overlapping character from given data. For example, for ‘GET/URI/’ 2-grams are: {‘GE’, ‘ET’, ‘T’, ‘/’, ‘/U’, ‘UR’, ‘RI’, ‘I/’}.

The size of each feature vector is equal to the number of all possible *n*-grams in the dataset. The *j*th feature of the *i*th feature vector x^{ij} is equal to the frequency of occurrences of the *j*th *n*-gram in the *i*th request.

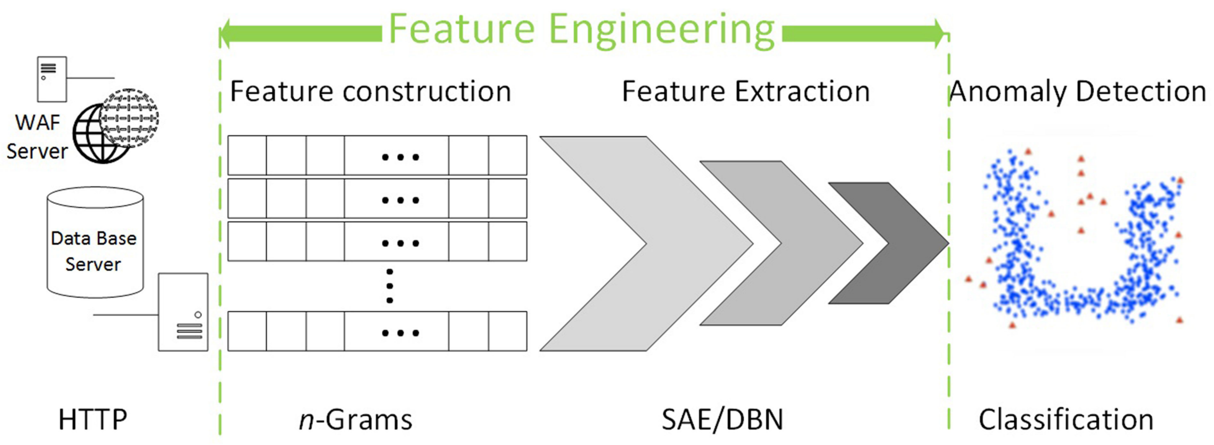

Fig. 1 *Scheme of the proposed anomaly detection method for WAF*

3.2 Feature extraction

The larger n for n-grams, the better modelling of data, but larger n needs larger data for modelling. Thus, there is a trade-off between bias and variance of the model [38]. Besides the growth of the grams, the size of sample space grows exponentially, and the problem quickly becomes ill-posed and intractable. To reduce the number of attributes, irrelevant features will be eliminated. Therefore, selection of relevant and essential features is very important. There are many methods, such as PCA and ICA, to extract features, but we are looking for approaches that do not miss any attributes, as well as non-linear combinations of features (so-called interferential features).

The performance and efficiency of machine learning algorithms highly depend on data representation [39]. Deep learning algorithms can make high-level and complex abstract features from data using a hierarchical learning process, i.e. representation learning [40]. In contrast to the previously proposed shallow architectures and kernel-based methods, deep learning models can learn good semantic features from their underlying training samples by applying discriminative models such as autoencoder and generative structures including restricted Boltzmann machine (RBM). Deep feature learning automatically extracts meaningful unsupervised features from the input data; thus, such approaches are not dependent on error-prone hand engineered features which can decrease the accuracy of the model. Moreover, deep architectures can capture highly abstracted mathematical features that are shared among various latent representation layers. This characteristic helps the network to share features which leads to the decrease of computational effort while yielding appropriate compactness for data representation.

In this paper, we applied two deep models namely SAE and DBN.

3.2.1 Stacked autoencoder: Assume the training examples in the form of $\{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$. An autoencoder neural network is an unsupervised learning algorithm that applies backpropagation, setting the target values to be equal to the inputs, i.e. $y^{(i)} = x^{(i)}$ for $i = 1 \ldots N$ [41]. An autoencoder always consists of two parts; the encoder f and the reconstruction part g as decoder. The encoder can be defined as $h = f(x)$ and reconstruction is defined as $r = g(h)$. In summary, the learning process is to find a value for parameter vector $\boldsymbol{\theta}$ to minimise reconstruction error as in the following equation:

$$J_{\text{AE}}(\boldsymbol{\theta}) = \sum_i L(x^{(i)}, r_\theta^{(i)}) \tag{1}$$

where L is a lost function that is used to compute error in neural networks, and one of the most widely used loss function is the mean square error, which calculates the square of difference between actual value and predicted value.

A SAE is a neural network consisting of multiple layers of autoencoders in which the outputs of hidden layer in each autoencoder are linked to the inputs of the next layer, i.e. the hidden neurons extract relevant features from the observations. These features can serve as input to another autoencoder. Also, the parameters $\boldsymbol{\theta}$ between layers are learnt layer-by-layer in each autoencoder.

3.2.2 Deep belief network: Like SAE, DBN is a stacked neural network but consists of multiple layers of RBM. RBM is an undirected probabilistic graphical model that describes probability distributions by mapping conditional dependence and independence properties between observable variables in a layer and latent variables in the other layer [42].

In the DBN with l layers, to model a joint distribution of the visible variable x and hidden layer h^k, we will have (2) as

$$\begin{aligned} &P(x, h^1, h^2, \ldots, h^l) \\ &= \left(\prod_{k=0}^{l-2} P(h^k \,|\, h^{k+1})\right) P(h^{l-1}, h^l), \quad x = h^0 \end{aligned} \tag{2}$$

where $P(h^k \,|\, h^{k+1})$ is the conditional distribution of the observed layer with the possession of the hidden layer and $P(h^{l-1}, h^l)$ is the joint distribution of available RBM at the highest level of a DBN [43].

3.3 One-class anomaly detection models

Once a stack of the neural network, DNN, has been built, its highest-level output representation can be used as input to a stand-alone supervised or unsupervised learning algorithm, for example, an SVM classifier.

The goal of anomaly detection is to separate the set of normal observations from some polluting ones, called ‘anomalies’. Anomalies are instances of data that do not conform to a well-defined notion of normal observations. We apply three algorithms to decide whether a new observation belongs to the same distribution as existing observations (normal) or should be considered as different (abnormal). All three learners have been used for one-class anomaly detection. These techniques use one-class learning techniques and in combination with feature extraction methods, learn a region that contains the normal-training data instances (train phase). Then, in the test phase, for each new instance, the machine determines if the new observation falls within the learned region or not.

3.3.1 One-class SVM: SVM is one of the most popular and powerful learning algorithms among machine learning community; it has different variations, one of them is 1SVM which is a supervised outlier detector and can be used in one-class applications such as anomaly detection [44]. The algorithm learns a decision function for normal data and classifies new data based on its similarity to the normal data.

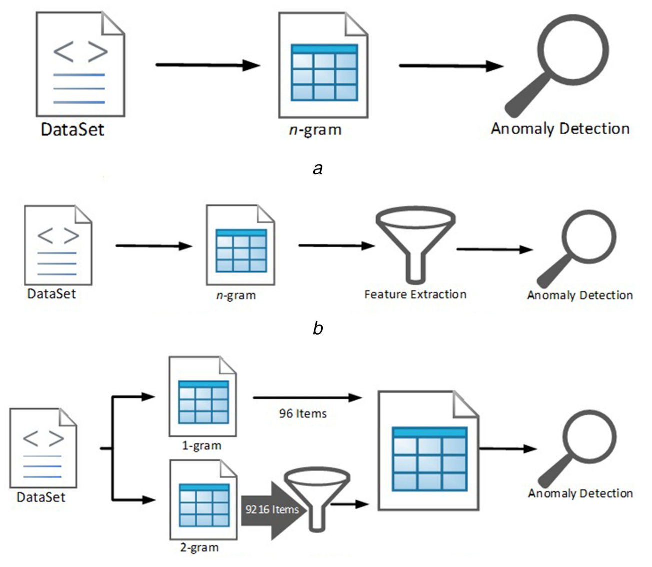

Fig. 2 *Different proposed scenarios*
(a) *n*-gram simple scenario, ***(b)*** Feature extraction scenario, ***(c)*** Feature fusion scenario

Table 1 Hardware specification of the system in benchmarking environment

CPU	RAM	OS
Intel Xeon 2.2 GHz (2 Processors)	64.0 GB	Windows 7 (64-bit)

1SVM separates all the data points from input feature space and maximises the distance from this hyperplane to this space. The quadratic programming minimisation functions as follows:

$$\min_{\omega,\,\varepsilon_i,\,\rho} \frac{1}{2}\parallel \omega \parallel^2 + \frac{1}{\vartheta n}\Sigma_{i=1}^{n}\varepsilon_i - \rho \qquad (3)$$

subject to

$$(\omega \cdot \varphi(x_i)) \geq \rho - \varepsilon_i \quad \text{for all } i = 1, \ldots, n$$
$$\varepsilon_i \geq 0 \quad \text{for all } i = 1, \ldots, n$$

$(\boldsymbol{\omega}, \boldsymbol{\rho})$ is a weight vector and an offset parametrising a hyperplane in the feature space associated with the kernel.

3.3.2 Isolation forest: Most model-based anomaly detection algorithms generate a model for normal, then mark data that do not conform the model as an anomaly. However, IF makes a model to 'isolate' anomalies from normal data explicitly. The idea behind the method is that since the anomalies are rare and different, they are more susceptible to isolation from normal data and this makes it suitable for anomaly detection applications such as WAF. This method partitions data using a random tree until all instances are isolated from each other [45]. IF is a two-stage process; first, it builds isolation trees using sub-samples of the training set and, later, it passes the test instances through isolation trees to obtain anomaly score for each instance. The time and memory complexity of an isolation tree grow linearly.

3.3.3 Elliptic envelope (Elliptic): Elliptic envelope is one of the anomaly detection algorithms; it assumes normal data came from a known distribution such as Gaussian distribution. The method tries to find the 'shape' of the data and mark data as abnormal which are far from the shape. As the name of the method shows, this method tries to fit an ellipse to the central data points using a robust covariance estimation to the data [46]. The objective of this method is to find observations whose covariance matrix has the lowest determinant. Also, it executes in an iteration process. In each process it computes and sorts distance for the data points, then, computes new distance and new covariance based on sorted distances. The algorithm stops when determinant of new covariance is equal to old covariance or zero.

3.4 Proposed scenarios

Once HTTP data is collected from WAF server to detect attacks based on machine learning, it is necessary to create features from requests inside the logs of data. In this work, we construct features based on *n*-grams. Also, we use one-class classifiers to detect anomalies. To evaluate the performance of feature extraction and feature fusion, according to Fig. 2 different scenarios are considered. In the first figure (Fig. 2*a*), we examine models without any feature extraction, then in Fig. 2*b* we can see how feature extraction stage is applied to detection models.

The fusion operation level can be divided into three types of information fusion namely: data fusion, feature fusion, and model fusion [47]. The aim of information fusion is to present a higher level of abstract of representation. The complementary information of two features acquired from 1-gram and 2-gram is useful for detection. We examine models in which DNN methods reduce the 2-gram features then these features are added to 1-gram features in parallel (Fig. 2*c*).

The next section is where we try to use three scenarios to evaluate our approach.

4 Simulation and evaluation

We conducted experiments on the CSIC 2010 dataset [48] and ECML/ PKDD 2007 dataset [49]. These datasets contain only HTTP traffic. CSIC is a publicly available labelled dataset. The training set contains 28,000 normal data and test set which contains 28,000 normal and 15,000 abnormal data. Also, the training set of ECML contains 20,000 normal data and test set contains 15,000 for each normal and abnormal data. Additionally, we examined experiments on a system with the following characteristics in Table 1.

Algorithms were implemented in Python 3.6 programming language and Tensorflow library [50]. Tensorflow is the machine learning framework that developed by Google and used to design, build, and train deep learning models.

Our method examines individual http requests. There is 256 ASCII code in total, but only 96 of them appear in HTTP dataset. The 1-gram model is computed as the frequency of occurrence of each ASCII code (96 characters) in the dataset. The 2-gram model is computed as two characters frequently. As a consequence, each HTTP request in 1-gram model is represented by a vector of 96 dimensions, while 2-grams feature vectors have 9216 dimensions, because we have 96 × 96 items.

To evaluate our deep model, we compare it with other feature extraction methods and the methods without feature extraction. We did experiments using three classifiers above. Accuracy, generalisation, and speed are more favourable metrics to evaluate our method. A fit generalisation is a good trade-off between false positive and false negative alarms [12]. We also used several measures to evaluate the performance of methods:

FP = False positive: the total number of attacks that do not detect,
FN = Falsenegative: the total number of normal requests that are classified as anomalous,
TP = True positive: the total number of attacks that are truly detected,
TN = True negative: the total number of normal requests that are classified as normal.

These measures are:

Accuracy (Acc.) = (TP + TN)/(TP + TN + FP + FN) × 100
Detection rate (DR) = Recall = True positive rate (TPR) = TP/(TP + FN) × 100
Precision (PR) = TP/(TP + FP) × 100
Specificity (Spec.) = TN/(FP + TN) × 100 = 100 − False positive rate (FPR)
F-score = *F*1 = (2 × PR × DR)/(PR + DR) × 100

Table 2 *n*-gram simple scenario (Fig. 2*a*) for CSIC data

Gram (dimension)	1-gram (96)			2-gram (9216)		
Measure	1-SVM	IF	Elliptic	1-SVM	IF	Elliptic
train time, s	63.82	10.26	2114.92	6165.30	1258.45	—
detect time, s	50.61	13.05	0.23	6296.33	708.95	—
Acc	79.68	77.23	64.17	82.70	72.48	out of memory
DR	79.69	54.27	34.93	88.53	40.00	
PR	67.88	73.77	48.39	70.00	68.28	
Spec	79.68	89.60	79.93	79.55	89.99	
*F*1	73.31	62.54	40.37	78.18	50.45	

Table 3 *n*-gram simple scenario (Fig. 2*a*) for ECML data

Gram (dimension) measure	1-gram (96)			2-gram (9216)		
	1-SVM	IF	Elliptic	1-SVM	IF	Elliptic
train time, s	342.03	5.88	—	18,177	855.07	—
detect time, s	129.33	7.59	—	13,745	491.44	—
Acc	50.18	78.17	single covariance matrix	50.18	61.53	single covariance matrix
DR	100	64.70		100	32.33	
PR	50.18	88.75		50.18	78.23	
Spec	0	91.74		0	90.94	
*F*1	66.82	74.84		66.82	45.76	

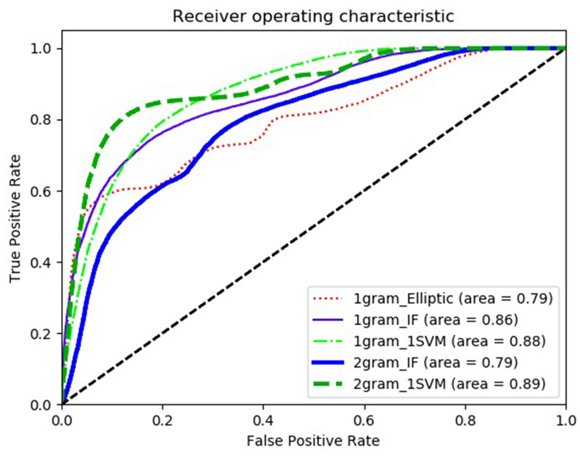

Fig. 3 *ROC curve for n-gram simple scenario (Fig. 2a) for CSIC data*

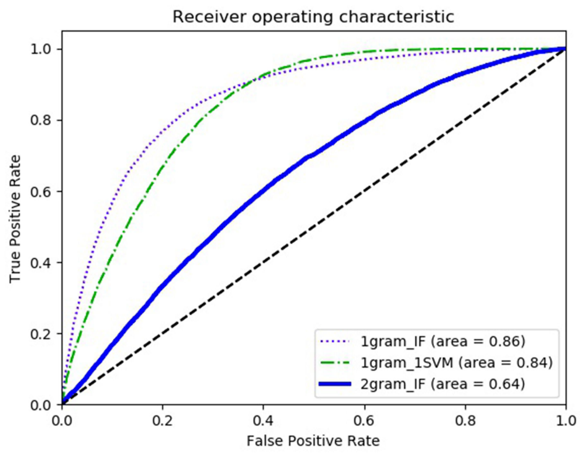

Fig. 4 *ROC curve for n-gram simple scenario (Fig. 2a) for ECML data*

ROC curve that provides insights closer to FPR and TPR for different thresholds of scores.

Tables 2 and 3 present the results of models which are implemented on three mentioned learners, i.e. SVM (1SVM), IF, and elliptic envelope (Elliptic) without feature extraction (*n*-gram simple scenario according to Fig. 2*a*). As we can see, generally, the speed of execution in training and specificity of IF is higher than others. Thus, the low value of the false positive rate of this method is remarkable. Also, IF can run on two models of the *n*-gram in a reasonable time, whereas, two other models, 1SVM and elliptic envelope could not execute on *n*-gram in ECML dataset (Table 3). As the results show, in 1SVM all normal and attack data are in one-class, i.e. the model cannot distinguish between normal and abnormal data. On the other hand, as we mentioned, the objective in the elliptic envelope is the lowest determinant. Thus, if determinant in each iteration becomes zero, the algorithm gives an error and stops. Also, elliptic envelope could not execute on *n*-gram with a higher order of *n* in CSIC dataset (Table 2). On the other hand, SVM has good accuracy and detection rate on 2-gram in CSIC data. Of course, we can increase the specificity of SVM with tuning penalty parameter but we know that the accuracy and detection rate decreases. Despite its performance, the detection time of elliptic envelope is noticeable. The detection time is important because the firewall is in a bottleneck. With these interpretations, although SVM offers the most accurate model for CSIC data and IF has good results, but in general, shows that the speed of execution and detection in 2-gram, increases the requirement of lesser and more effective features. Also, despite the fact that the 2-gram method should work better but has not been able to perform satisfactorily at all measures, it seems the feature extraction is essential for the 2-gram method. The ROC curves of these models are shown in Figs. 3 and 4 for CSIC and ECML datasets, respectively.

According to Fig. 2*b*, once we use *n*-gram with a higher order of *n*, reduction of dimensions is inevitable. However, it is necessary to use feature extraction models to extract relevant attributes and execute in reasonable time. Before we demonstrate the efficiency and effectiveness of deep learning models, we illustrate results of models in which features are extracted by PCA, non-linear or KPCA and FICA in Tables 4 and 5 (feature extraction scenario). In our implementations, the kernel used in KPCA is RBF, and also, we can see it is time-consuming and slow. The dimensions of 2-gram models reduced into 100 dimensions. In addition to runtime, what is remarkable is that false alarm rate of IF is lower than others and accuracy of none of them is noticeable. Although these methods did not improve the accuracy of the detection, the good of these methods, as before, is that these methods can be implemented for any *n*-gram model. On the other hand, because the dimensions of models have been reduced to 100 and the detection time is less, this should also be considered as the time of transformation (Trans.). Feature extraction must be added to detection time. For this purpose, the transform time of test data

Table 4 Feature extraction scenario (Fig. 2*b*) including PCA, KPCA, and FICA for CSIC data

Feature extraction		PCA			KPCA			FICA		
times, s	train		118.35			1725.64			372.81	
	trans.		46.45			501.28			51.55	
Measure		I-SVM	IF	Elliptic	I-SVM	IF	Elliptic	I-SVM	IF	Elliptic
train time, s		64.63	12.16	1457.97	62.19	9.81	1473.61	63.17	9.24	1361.88
detect time, s		52.96	10.92	0.23	52.73	11.58	0.27	52.48	11.06	0.23
Acc		78.28	77.43	71.88	69.25	76.92	72.46	75.41	76.00	72.06
DR		64.51	49.20	56.50	45.52	45.91	58.58	62.50	44.43	57.32
PR		70.86	78.26	60.55	57.72	79.51	61.15	65.64	77.39	60.70
Spec		85.70	92.64	80.17	82.04	93.62	79.94	82.37	93.01	80.00
*F*1		67.53	60.42	58.45	50.90	58.21	59.84	64.03	56.45	58.96

Table 5 Feature extraction scenario (Fig. 2*b*) including PCA, KPCA and FICA for ECML data

Feature extraction		PCA			KPCA			FICA		
times, s	train		21.70			506.03			304.04	
	trans.		2.55			66.46			2.82	
Measure		I-SVM	IF	Elliptic	I-SVM	IF	Elliptic	I-SVM	IF	Elliptic
train time, s		181.71	7.61	1096.73	31.96	6.80	1319.09	32.68	6.37	1096.01
detect time, s		105.45	7.53	0.19	26.36	7.77	0.17	26.34	7.83	0.17
Acc		59.52	68.61	50.94	56.57	63.99	49.92	70.75	64.91	50.78
DR		88.66	42.50	16.65	29.50	32.33	15.18	51.41	35.77	16.30
PR		56.11	89.38	53.59	64.76	88.75	50.31	84.06	86.25	53.12
Spec		30.17	94.92	85.48	83.83	95.87	84.90	90.17	94.26	85.51
*F*1		68.73	57.61	25.41	40.54	47.39	23.32	63.84	50.27	24.94

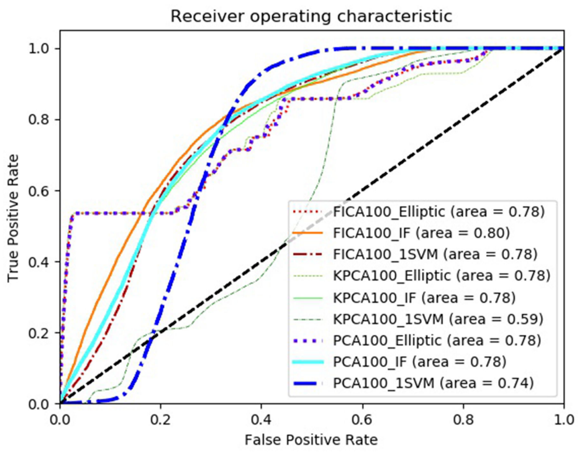

Fig. 5 *ROC curve for feature extraction scenario (Fig. 2b) including PCA, KPCA, and FICA for CSIC data*

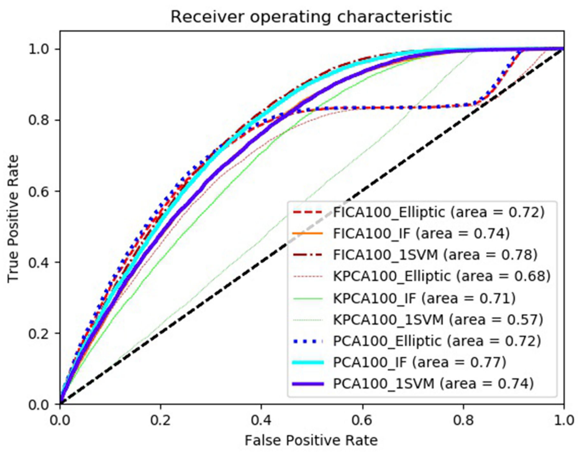

Fig. 6 *ROC curve for feature extraction scenario (Fig. 2b) including PCA, KPCA, and FICA for ECML data*

must be added to the detection time. The area under curve (AUC) in Figs. 5 and 6 demonstrates that these models do not have better sensitivity and specificity than simple scenarios.

In fact, CSIC dataset has been observed to have many features that were linearly correlated to each other, where in the ECML dataset, the non-linear relations between features were more representative [16]. According to Fig. 2*b* using DNN feature extraction, as we see in Tables 6 and 7, SAE as the discriminative model is better than DBN as a generative model for CSIC data, but this is the opposite for ECML data. Tables 6 and 7 and ROC curves in Figs. 7 and 8 show that IF has better results than others, and it is best in CSIC data using SAE model. As we said, the transformation time (Trans.) of test data in these models is added to detection time. The transformation time in deep models is based on the complexity of structures. Once we select greater structure, the transformation time would be greater too.

The structure of stacked DNN for 2-gram models are 1000, 400, and 100 hidden neurons, respectively. Thus, we have several models that extract 100 features from data, afterwards, three learners examine efficiency and effectiveness of these models.

According to Fig. 2*c* using parallel feature fusion, as we see in Tables 8 and 9, a fusion of features improved the measures of models because we applied extra information of samples specification. These results are also better in ROC curve in Figs. 9 and 10. As we see, the AUC of model is more than 80% for all manners. Thus, the accuracy and generalisation is improved in feature fusion scenarios. Of course, the number of attributes also increases due to the interference of two sets of features, but the speed of execution of the models can be said to be reasonable. Also, when DNN models added one layer to reduce 100 dimensions to 30 dimensions plus using 1-gram, the SAE and IF detection model would have better results compared to 100 dimensions (as shown in Fig. 2*c*). We can see the results in Table 10 and Fig. 11. Also, elliptic envelope detection model has good results on ECML data, when feature fusion models are used.

5 Discussion

In this section, we interpret the experiments and explore their significance, focusing on the accuracy, generalisation, and speed of our methods.

Table 6 Feature extraction scenario (Fig. 2*b*) including deep methods for CSIC data

Deep structure		SAE [4000, 1000, 400, 100]			DBN [4000, 1000, 400, 100]		
times, s	train		2632.49			3315.16	
	trans.		29.34			25.52	
Measure		1-SVM	IF	Elliptic	1-SVM	IF	Elliptic
train time, s		62.43	7.93	1526.88	25.96	5.75	830.00
detect time, s		51.22	8.55	0.24	49.89	6.52	0.33
Acc		70.20	87.20	65.43	38.41	80.21	62.57
DR		47.02	85.46	38.81	60.82	62.54	30.40
PR		59.40	79.51	50.83	30.79	76.66	44.91
Spec		82.68	88.13	79.77	26.33	89.74	79.90
*F*1		52.49	82.38	44.01	40.88	68.88	36.26

Table 7 Feature extraction scenario (Fig. 2*b*) including deep methods for ECML data

Deep structure		SAE [4000, 1000, 400, 100]			DBN [4000, 1000, 400, 100]		
times, s	train		1387.45			1783.91	
	trans.		10.91			10.41	
Measure		1-SVM	IF	Elliptic	1-SVM	IF	Elliptic
train time, s		12.81	4.43	579.43	9.58	3.99	551.80
detect time, s		26.90	4.60	0.17	22.01	4.25	0.18
Acc		60.25	62.72	49.93	50.18	64.67	58.81
DR		48.24	36.11	22.33	100	40.22	40.21
PR		63.73	77.62	50.24	50.18	79.10	64.32
Spec		72.36	89.52	77.73	0	89.30	77.54
*F*1		54.91	49.29	30.92	66.82	53.33	49.48

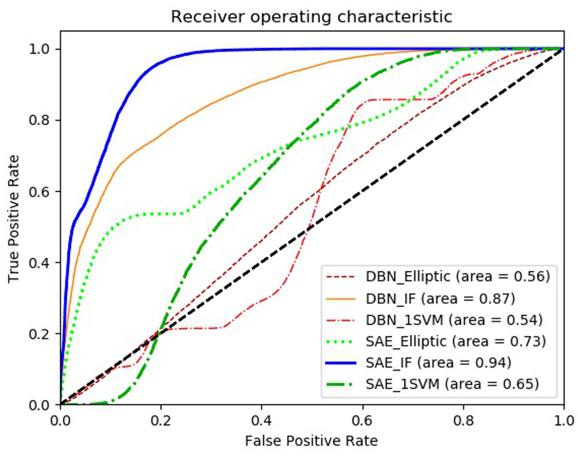

Fig. 7 *ROC curve for feature extraction scenario (Fig. 2b) including deep methods for CSIC data*

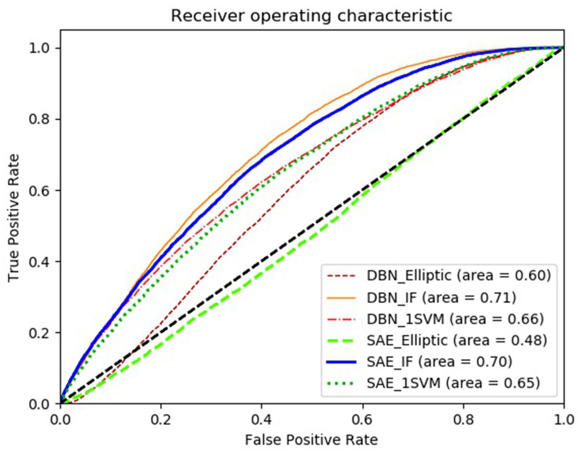

Fig. 8 *ROC curve for feature extraction scenario (Fig. 2b) including deep methods for ECML data*

Table 8 Feature fusion scenario (Fig. 2*c*) including deep methods over 2-gram and 1-gram for CSIC data

Deep structure	SAE			DBN		
	[4000, 1000, 400, 100] + 1-gram (96)			[4000, 1000, 400, 100] + 1-gram (96)		
Measure	1-SVM	IF	Elliptic	1-SVM	IF	Elliptic
train time, s	142.84	22.67	422.73	127.7	23.31	416.33
detect time, s	96.19	19.94	0.55	95.87	19.46	0.49
Acc	79.72	88.60	61.84	79.68	86.87	72.70
DR	79.74	88.09	62.36	79.69	81.53	59.21
PR	67.92	81.01	79.88	67.88	81.09	61.43
Spec	79.71	88.88	62.10	79.68	89.75	79.97
*F*1	73.36	84.40	0.83	73.31	81.31	60.30

Table 9 Feature fusion scenario (Fig. 2*c*) including deep methods over 2-gram and 1-gram for ECML data

Deep structure	SAE			DBN		
	[4000, 1000, 400, 100] + 1-gram (96)			[4000, 1000, 400, 100] + 1-gram (96)		
Measure	1-SVM	IF	Elliptic	1-SVM	IF	Elliptic
train time, s	57.58	19.62	298.55	55.89	18.58	301.19
detect time, s	49.79	16.82	0.51	50.05	16.99	0.41
Acc	74.40	68.95	83.78	74.40	71.98	84.13
DR	68.78	48.48	86.37	68.78	55.11	89.56
PR	77.65	82.40	82.40	77.65	83.42	80.86
Spec	80.06	89.57	81.17	80.06	88.97	78.65
*F*1	72.95	61.04	84.24	72.95	66.37	84.99

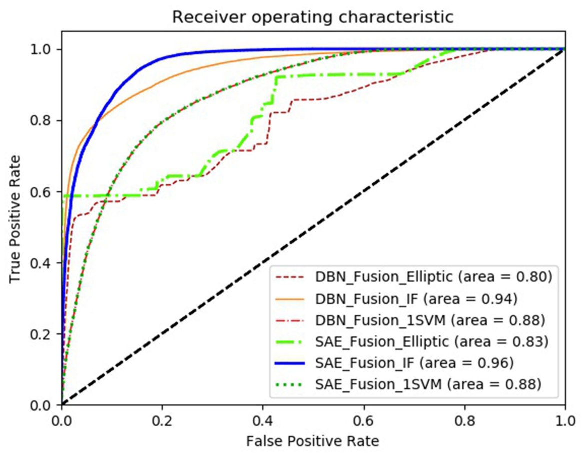

Fig. 9 *ROC curve for feature fusion scenario (Fig. 2c) including 1-gram and deep methods over 2-gram for CSIC data*

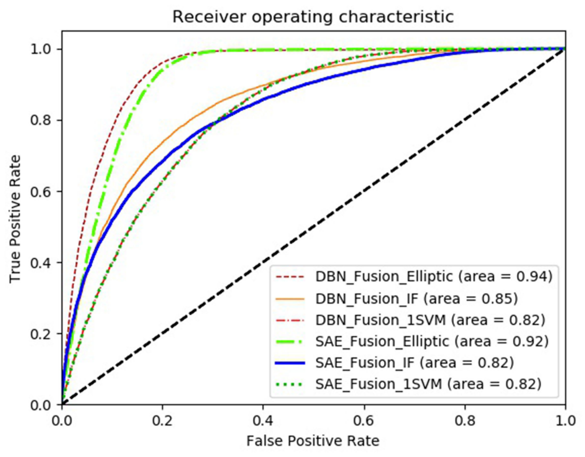

Fig. 10 *ROC curve for feature fusion scenario (Fig. 2c) including 1-gram and deep methods over 2-gram for ECML data*

Web application request data becomes increasingly massive, therefore, building a WAF model to efficient and effective detection of abnormal behaviour becomes an important challenge. *N*-gram model is a language-independent statistical model to retrieve information from HTTP requests within our proposed methods. To extract relevant and essential features from *n*-gram vector, we applied DNNs. There are also limitations in our work; the most important is that deep models can be time consuming in training for higher orders of *n*-grams. However, we must trade off between speed, accuracy, and generalisation. On the other hand, deep models are practically one of the best options for large and high dimensional data. To examine proposed methods, we demonstrated the comparison of the performance of models in bar charts of Figs. 12 and 13.

Fig. 12 shows the accuracy of models which have accuracy more than 80%. As the figure shows, among those models not using deep learning, only the accuracy of 1SVM on 2-gram and CSIC data has a value higher than 80%. But if we do not consider training time of this model, the detection time of this is bad. As we know, the training phase performs offline, but the detection phase performs online. Therefore, in order not to have a lot of overhead on the bottleneck, it is to decrease the time of detection. Also, the accuracy values of deep models especially based on feature fusion are remarkable. As seen in Fig. 12, the CSIC data is more accurate with IF models, and the ECML data which has more non-linearity relations between features is more accurate with elliptic envelope models, both of which are used in feature fusion scenario.

The ROC curve and the area under the area of ROC is a good measure for investigating generalisation of different methods. A good generalisation is a proper trade-off between FNR and TPR. For this purpose, minimising the impact of attacks on data is essential to minimise generalisation. Thus, both of the values of detection rate and specificity need to be good.

Fig. 13 depicts models that have AUC higher than 0.9, along with the two other measures, the detection rate and specificity values. As we see, only the models using deep learning especially the feature fusion have a good generalisation. Also, the CSIC data and ECML data are more suitable generalisations with IF models and elliptic envelope models used in feature fusion scenario, respectively.

As we observe the results, we can see 1SVM methods are time consuming, and the methods based on IF are fast, and the detection time of methods based on the elliptic envelope which is very good but the training time of them is very time consuming. Also, the elliptic envelope is sensitive to matrix covariance of data. In addition to these cases, we need to perform detection very fast. For this purpose, it is necessary that the number of features of machine

Table 10 Feature fusion scenario (Fig. 2*c*) including deep methods over 2-gram are reduced to 30 and 1-gram

Deep structure	SAE		DBN	
	[4000, 1000, 400, 100, 30] + 1-gram (96)			
Data	CSIC	ECML	ECML	
Measure	IF	IF	Elliptic	Elliptic
train time, s	14.14	13.89	286.55	284.54
detect time, s	14.05	13.97	0.25	0.25
Acc	89.24	77.47	83.41	84.02
DR	89.48	64.88	88.74	89.75
PR	81.58	86.9	80.28	80.61
Spec	89.11	90.15	78.05	78.25
*F*1	85.35	74.29	84.30	84.93

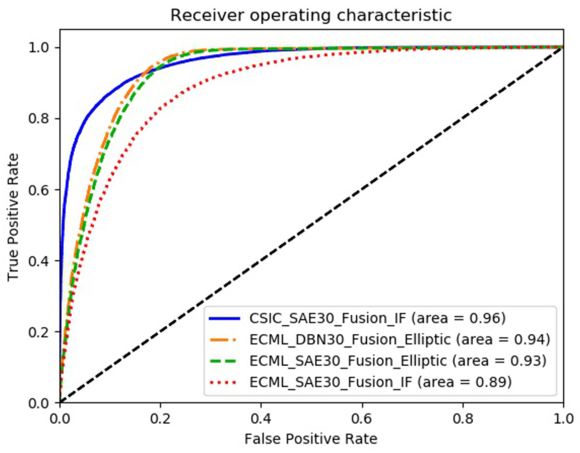

Fig. 11 *ROC curve for feature fusion scenario (Fig. 2c) including deep methods over 2-gram are reduced to 30 and 1-gram*

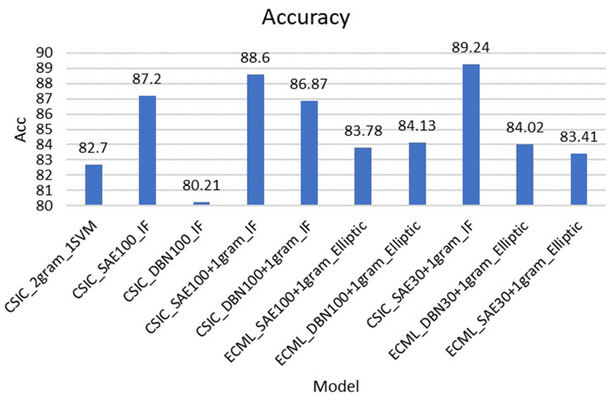

Fig. 12 *Accuracy of models which have high accuracy*

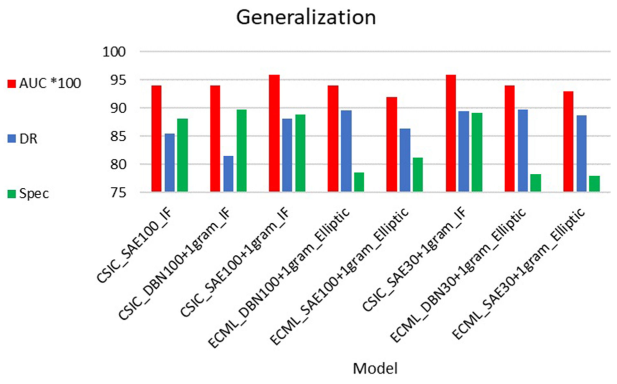

Fig. 13 *Generalisation (AUC, detection rate, and specificity) of models*

learning should not be high. Therefore, the feature extraction is inevitable. However, models based on DNNs especially parallel feature fusions are optimum methods in the trade-off between generalisation, accuracy, and speed.

Researches and methods, which are mentioned in related works, are either supervised or unsupervised. In many of these works tokenised items such as method type, famous headers like language header and OS header constructs the feature vector. Also, the methods like neural networks, Markov-based and clustering algorithms are developed for attacks detection, while we construct features from *n*-gram character and then apply semi-supervised methods, i.e. one-class algorithm for detection.

6 Conclusions and future works

In this study, we have leveraged DNN methods to extract relevant features from HTTP request logs for anomaly detection in WAFs. *N*-gram based on character is a method that we used to construct features from our data. Also, SAE and DBN are used as DNN methods, which are applied to 2-gram features, then, we fused them with 1-gram features in parallel to robust resulting feature sets for anomaly detection purposes. We applied our methods on CSIC 2010 and ECML/PKDD 2007 datasets. 1SVM, elliptic envelope, and IF have been implemented as one-class learner to distinguish normal data from abnormal ones. The results show that in contrast to the traditional models, deep models especially fusion models have better performance regarding accuracy and generalisation in reasonable detection time.

For future works, we can consider the use of another feature set with various structures with our feature fusion scenario model to extract effective features for anomaly detection models. Also, an extension of the method for big data environments and data streams can be considered. Porting the WAF system as a cloud service is also another suggestion for future works.

7 References

[1] Liao, H.-J., Lin, C.-H.R., Lin, Y.-C., *et al.*: 'Intrusion detection system: a comprehensive review', *J. Netw. Comput. Appl.*, 2013, **36**, (1), pp. 16–24
[2] Chandola, V., Banerjee, A., Kumar, V.: 'Anomaly detection: a survey', *ACM Comput. Surv.*, 2009, **41**, (3), p. 15
[3] Nguyen, H.T., Franke, K., Petrovic, S.: 'Feature extraction methods for intrusion detection systems', *Threats Countermeas. Adv. Appl. Inf. Secur.*, 2012, **3**, pp. 23–52
[4] Wang, K., Stolfo, S.J.: 'Anomalous payload-based network intrusion detection'. Int. Workshop on Recent Advances in Intrusion Detection, Springer, Berlin, Heidelberg, 2004, pp. 203–222
[5] Wang, K., Parekh, J., Stolfo, S.: 'Anagram: a content anomaly detector resistant to mimicry attack'. Int. Workshop on Recent Advances in Intrusion Detection, Springer, Berlin, Heidelberg, 2006, pp. 226–248
[6] Perdisci, R., Ariu, D., Fogla, P., *et al.*: 'McPAD: a multiple classifier system for accurate payload-based anomaly detection', *Comput. Netw.*, 2009, **53**, (6), pp. 864–881
[7] Ma, T., Wang, F., Cheng, J., *et al.*: 'A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks', *Sensors*, 2016, **16**, (10), p. 1701
[8] Niyaz, Q., Sun, W., Javaid, A.Y.: 'A deep learning based DDoS detection system in software-defined networking (SDN)', arXiv preprint arXiv:1611.07400, 2016
[9] Kruegel, C., Vigna, G.: 'Anomaly detection of web-based attacks'. Proc. of the 10th ACM Conf. on Computer and Communications Security, ACM, Singapore, Republic of Singapore, 2003, pp. 251–261
[10] Kruegel, C., Vigna, G., Robertson, W.: 'A multi-model approach to the detection of web-based attacks', *Comput. Netw.*, 2005, **48**, (5), pp. 717–738
[11] Ingham, K.L., Inoue, H.: 'Comparing anomaly detection techniques for HTTP'. In Int. Workshop on Recent Advances in Intrusion Detection, Springer, Berlin, Heidelberg, 2007, pp. 42–62

[12] Ingham, K.L., Somayaji, A., Burge, J., *et al.*: 'Learning DFA representations of HTTP for protecting web applications', *Comput. Netw.*, 2007, **51**, (5), pp. 1239–1255

[13] Song, Y., Keromytis, A.D., Stolfo, S.J.: 'Spectrogram: a mixture-of-Markov-chains model for anomaly detection in web traffic'. Proc. of the Network and Distributed System Security Symp., NDSS, NDSS, San Diego, California, 2009, vol. 9, pp. 1–15

[14] Zolotukhin, M., Hamalainen, T., Kokkonen, T., *et al.*: 'Analysis of HTTP requests for anomaly detection of web attacks'. 2014 IEEE 12th Int. Conf. on Dependable, Autonomic and Secure Computing (DASC), IEEE, Dalian, China, 2014, pp. 406–411

[15] Jamdagni, A., Tan, Z., Nanda, P., *et al.*: 'Intrusion detection using GSAD model for HTTP traffic on web services'. Proc. of the 6th Int. Wireless Communications and Mobile Computing Conf., ACM, Caen, France, 2010, pp. 1193–1197

[16] Nguyen, H.T., Torrano-Gimenez, C., Alvarez, G., *et al.*: 'Enhancing the effectiveness of web application firewalls by generic feature selection', *Log. J. IGPL*, 2012, **21**, (4), pp. 560–570

[17] Torrano-Gimenez, C., Nguyen, H.T., Alvarez, G., *et al.*: 'Combining expert knowledge with automatic feature extraction for reliable web attack detection', *Secur. Commun. Netw.*, 2015, **8**, (16), pp. 2750–2767

[18] Torrano-Gimenez, C., Nguyen, H.T., Alvarez, G., *et al.*: 'Applying feature selection to payload-based web application firewalls'. 2011 Third Int. Workshop on Security and Communication Networks (IWSCN), IEEE, Banff, AB, Canada, 2011, pp. 75–81

[19] Wang, W., Zhang, X.: 'High-speed web attack detection through extracting exemplars from HTTP traffic'. Proc. of the 2011 ACM Symp. on Applied Computing, ACM, Taichung, Taiwan, 2011, pp. 1538–1543

[20] Wang, W., Guyet, T., Quiniou, R., *et al.*: 'Autonomic intrusion detection: adaptively detecting anomalies over unlabeled audit data streams in computer networks', *Knowl.-Based Syst.*, 2014, **70**, pp. 103–117

[21] Cappo, C., Nunes, R.C., Schaerer, C.: 'On using wavelets for detecting attacks to web-based applications'. CONGRESSO NACIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL, XXXII, Cuiabá, 2009. Proc. Cuiabá, CNMAC, CNMAC, Cuiabá, Brazil, 2009, pp. 1040–1041

[22] Mozzaquatro, B.A., de Azevedo, R.P., Nunes, R.C., *et al.*: 'Anomaly-based techniques for web attacks detection', *J. Appl. Comput. Res.*, 2011, **1**, (2), pp. 111–120

[23] Stephan, J.J., Mohammed, S.D., Abbas, M.K.: 'Neural network approach to web application protection', *Int. J. Inf. Educ. Technol.*, 2015, **5**, (2), p. 150

[24] Singh, K.J., Thongam, K., De, T.: 'Detection and differentiation of application layer DDoS attack from flash events using fuzzy-GA computation', *IET Inf. Sec.*, 2018, **12**, (6), pp. 502–512

[25] Johnson Singh, K., Thongam, K., De, T.: 'Entropy-based application layer DDoS attack detection using artificial neural networks', *Entropy*, 2016, **18**, (10), p. 350

[26] Shalaginov, A., Franke, K.: 'Multinomial classification of web attacks using improved fuzzy rules learning by neuro-fuzzy', *Int. J. Hybrid. Intell. Syst.*, 2016, **13**, (1), pp. 15–26

[27] Kozik, R., Choras, M., Holubowicz, W., *et al.*: 'Extreme learning machines for web layer anomaly detection'. Int. Conf. on Image Processing and Communications, Springer, Bydgoszcz, Poland, 2016, pp. 226–233

[28] Estevez-Tapiador, J.M., Garcia-Teodoro, P., Diaz-Verdejo, J.E.: 'Detection of web-based attacks through Markovian protocol parsing'. 10th IEEE Symp. on Computers and Communications, 2005. ISCC 2005. Proc., IEEE, Murcia, Spain, 2005, pp. 457–462

[29] Relan, V., Sonawane, B.: 'Detection and mitigation of web services attacks using Markov model'. CMSC 678: Machine Learning Project, CMSC, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 2009

[30] Lampesberger, H., Winter, P., Zeilinger, M., *et al.*: 'An on-line learning statistical model to detect malicious web requests'. Int. Conf. on Security and Privacy in Communication Systems, Springer, Berlin, Heidelberg, 2011, pp. 19–38

[31] Zhang, M., Lu, S., Xu, B.: 'An anomaly detection method based on multi-models to detect web attacks'. 2017 10th Int. Symp. on Computational Intelligence and Design (ISCID), ISCID, Hangzhou, China, 2017, pp. 404–409

[32] Razzaq, A., Anwar, Z., Ahmad, H.F., *et al.*: 'Ontology for attack detection: an intelligent approach to web application security', *Comput. Secur.*, 2014, **45**, pp. 124–146

[33] Razzaq, A., Latif, K., Ahmad, H.F., *et al.*: 'Semantic security against web application attacks', *Inf. Sci.*, 2014, **254**, pp. 19–38

[34] Vartouni, A.M., Kashi, S.S., Teshnehlab, M.: 'An anomaly detection method to detect web attacks using stacked auto-encoder'. 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), CFIS, Kerman, Iran, 2018, pp. 131–134

[35] Pan, Y., Sun, F., White, J., *et al.*: '*Detecting web attacks with end-to-end deep learning*', (Vanderbilt University, Melbourne, FL, USA, 2018), pp. 1–14

[36] Liang, J., Zhao, W., Ye, W.: 'Anomaly-based web attack detection: a deep learning approach'. Int. Conf. on Network, Communication and Computing, ACM, Kunming, China, 2017, pp. 80–85

[37] Zhang, M., Xu, B., Bai, S., *et al.*: 'A deep learning method to detect web attacks using a specially designed CNN'. Int. Conf. on Neural Information Processing, Springer, Guangzhou, China, 2017, pp. 828–836

[38] Jurafsky, D., Martin, J.H.: '*Speech and language processing*' (London Pearson, Stanford University, USA, 2014, 2nd edn. 2018)

[39] Bengio, Y., Courville, A., Vincent, P.: 'Representation learning: a review and new perspectives', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2013, **35**, (8), pp. 1798–1828

[40] Najafabadi, M.M., Villanustre, F., Khoshgoftaar, T.M., *et al.*: 'Deep learning applications and challenges in big data analytics', *J. Big. Data.*, 2015, **2**, (1), p. 1

[41] Goodfellow, I., Bengio, Y., Courville, A.: '*Deep learning*' (MIT Press, Cambridge, Massachusetts, USA, 2016)

[42] Fischer, A., Igel, C.: 'An introduction to restricted Boltzmann machines'. Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, Springer, Buenos Aires, Argentina, 2012, pp. 14–36

[43] Hinton, G.E.: 'Deep belief networks', *Scholarpedia*, 2009, **4**, (5), p. 5947

[44] Schölkopf, B., Platt, J. C., Shawe-Taylor, J., *et al.*: 'Estimating the support of a high-dimensional distribution', *Neural Comput.*, 2001, **13**, (7), pp. 1443–1471

[45] Liu, F.T., Ting, K.M., Zhou, Z.-H.: 'Isolation forest'. Eighth IEEE Int. Conf. on Data Mining, ICDM, Pisa, Italy, 2008, pp. 413–422

[46] Rousseeuw, P.J., Driessen, K.V.: 'A fast algorithm for the minimum covariance determinant estimator', *Technometrics*, 1999, **41**, (3), pp. 212–223

[47] Chen, Z., Li, W.: 'Multisensor feature fusion for bearing fault diagnosis using sparse autoencoder and deep belief network', *IEEE Trans. Instrum. Meas.*, 2017, **66**, (7), pp. 1693–1702

[48] Torrano-Gimnez, C., Prez-Villegas, A., Alvarez, G.: 'The HTTP dataset CSIC 2010', Instituto de Seguridad de la Informacion (ISI), 2010

[49] Raissi, C., Brissaud, J., Dray, G., *et al.*: 'Web analyzing traffic challenge: description and results'. Proc. of the ECML/PKDD, Springer, Warsaw, Poland, 2007, pp. 47–52

[50] Abadi, M., Agarwal, A., Barham, P., *et al.*: 'Tensorflow: large-scale machine learning on heterogeneous distributed systems', arXiv preprint arXiv:1603.04467, 2016