A Novel Data Augmentation Technique and Deep Learning Model for Web Application Security

HACER KARACAN†1, (Member, IEEE), AND MEHMET SEVRI‡2, (Graduate Student Member, IEEE)
1Computer Engineering Department, Faculty of Engineering, Gazi University, 06570 Ankara, Turkey
2Informatics Institute, Gazi University, 06680 Ankara, Turkey
Corresponding author: Mehmet Sevri (mehmetsevri@gmail.com)

ABSTRACT Web applications are often exposed to attacks because of the critical information and valuable assets they host. In this study, Bi-LSTM based web application security models were developed in order to detect web attacks and classify them into binary or multiple classes using HTTP requests. A novel data augmentation technique based on the self-adapting noise adding method (DA-SANA) was developed. The DA-SANA method solves the low sensitivity problem caused by imbalanced data and the complex structure of multi-class classification in web attack detection. Experimental evaluations are carried out in detail using two benchmark web security datasets and a newly created dataset within the scope of the study. The achieved worst case detection rates are 98.34% and 93.91% for binary-class and multi-class classifications, respectively. The proposed DA-SANA technique provides an average of 6.52% improvement in multi-class classification for two datasets. These results revealed that the best classification performance values were achieved when compared with previous studies.

INDEX TERMS Web security, anomaly detection, deep learning, Bi-LSTM, data augmentation.

I. INTRODUCTION
Significant attacks such as SQL injection (SQLi), cross-site scripting (XSS), remote code execution (RCE), local file inclusion (LFI), broken authentication, sensitive data exposure, XML external entities (XXE), and cross-site request forgery (CSRF) can be performed against web applications. Because of these web attacks, critical data can be exposed, systems can be hijacked, and significant privacy violations or financial losses can occur. Since many economic and critical transactions are carried out through web applications, the number of hackers attacking these applications is increasing daily, with the attacks becoming more sophisticated. Even though the tools used in the attacks have greatly improved over time and have supported more sophisticated attacks; the level of technical knowledge and skills required to perform an attack has decreased. Therewithal, even very sophisticated and damaging attacks can be carried out by “just curious or lamer” individuals with exiguous technical knowledge [1].

The top ten of the most critical security risks for web applications was published by Owasp [2]. According to the Owasp report, the most dangerous of these security risks are injection attacks and the best-known are SQLi, XSS and command injection attacks. Using these injection vulnerabilities, it is possible to leak critical information or run the application unexpectedly by executing unwanted codes on the web server or user browser by using attacker-supplied data that are non-validated, non-filtered or non-sanitized.

In traditional firewalls, OSI level 4 security rules are created based on source IP, destination IP, port number and packet type (TCP or UDP). Since a web application communicates with end-users on TCP 80 (HTTP) or 443 (HTTPS) port, traffics must be allowed from these ports. In this case, a vulnerability in the web application can be easily exploited without being noticed by traditional firewalls. Web application firewalls (WAF) should be used to prevent exploitation of the vulnerabilities found in web applications. WAF systems can deeply analyze web requests in more detail, detect and prevent web attacks, and generate alarms including the details of the attacks [3]. There are two main methods, signature-based and anomaly-based, in detecting web attacks. There are some serious challenges to overcome in signature-based systems. Signature-based systems cannot detect up-to-date attack types. In addition, they can be easily bypassed with different encoding and bypass techniques [4]. Furthermore, the high false-positive (FP) ratio is
another critical challenge because of the strict security rules in signature-based WAF systems. Anomaly-based WAFs are an effective method to overcome these problems. Robust WAF models can be developed that can successfully detect attacks including up-to-date attack types through features based on web requests.

### A. PURPOSE AND MOTIVATIONS

In this study, effective WAF models based on deep learning (DL) were developed to successfully classify binary-class and multi-class attacks to web applications. This development process was carried out in two stages. In the first stage, after the text-based HTTP requests were processed through word tokenization, data augmentation was performed using the DA-SANA method developed in this study. In the second stage, binary-class and multi-class web attack detection models based on different DL architectures were developed. Within the scope of the study, Convolutional Neural Network (CNN), Deep Neural Network (DNN), Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU), and Bidirectional Long Short Term Memory (Bi-LSTM) architectures were used for the development of the WAF models. The performances of the models were evaluated using three different datasets, one of which was created within the scope of the study. The remaining two are the most frequently used benchmark datasets in the web security field, and our results are compared with previous studies. The three most important motivations for conducting this study are;

1. Web application security, as described above, is critical and contains urgent and outstanding problems. One of the important challenges is that attack techniques are increasing day by day, thus there is need for up-to-date solution approaches.
2. As explained above, most of the studies in the security field are network-based security studies. There is a lack of research studies in the field of web security.
3. The impressive success of deep learning methods in similar areas of study was an important source of motivation when determining the method of this study.

### B. CONTRIBUTIONS

The main contributions of this study are given below;

1. A newly up-to-date and robust HTTP web attack dataset was created with this study.
2. A novel data augmentation technique was proposed as the DA-SANA. In the DA-SANA method, the noise coefficients were determined based on the statistically distinctive features of the attack requests, as described in detail later in this paper. With the adaptive noise addition, normal and abnormal HTTP traffics were separated from each other in hidden space, thus improving the detection rates.
3. Novel WAF models, that can work successfully in real-time, were developed for web attack detection. To the best of our knowledge, performance evaluations in the study stand out as the best performance values in the literature for both binary-class and multi-class.
4. The codes and pre-processed datasets used in this study are publicly shared for reproducibility and as a contribution to similar works in the future*

*https://github.com/mehmetsevri/Deep-Learning-based-WAF

The remainder of the study is organized as follows. The second section, related studies, presents similar studies in the field of web security. In the third section, the preliminaries of feature construction and Bi-LSTM, which was the most successful model, are provided through mathematical representations of the structures used in the detection of the web attacks. Section 4 presents the methodology, model designs, and definitions for the classification models used classification models used in the study. In section 5, the experimental design, descriptions of datasets and validation methods are explained. Section 6, discusses the results related to the experimental performance evaluations of the datasets and examines comparisons to previous studies. Finally, the conclusion and possible future works are given in section 7.

### II. RELATED WORK

A look at current machine learning based cybersecurity studies, shows more studies focused on network security and malware detection than web security. There are some web application security studies that use different traditional machine learning algorithms and deep learning methods. In this section, we briefly introduce and discuss some important web security studies based on traditional machine learning and deep learning algorithms.

#### A. TRADITIONAL MACHINE LEARNING METHODS

Anomaly-based web intrusion detection is an important area of focus in web security research. One of the first studies was done by Krueger and Vigna in 2003, using Markov models. The study remains important because it is the first study on anomaly-based web attack detection in the literature and it concentrates on the extraction of the features that used in web attacks.

Liu et al. [6] created a web intrusion detection system based on feature analysis and the SVM algorithm, using statistical and manual techniques for feature extraction. The model achieved a 95.8% detection rate on the CSIC-2010 dataset. The A-ExIDS is proposed in this study [7] as an adaptive ensemble WAF model based on four classification models: Naive Bayes, Bayes Network, Decision Stump and RBFNetwork. Thang [8] developed a WAF that can detect injection attacks (SQLi, XSS, etc.) by filtering web requests via regular expressions and then classify them using a trained Random Forest (RF) model on the CSIC-2010 dataset. Zhou and Wang [9] designed an ensemble model combining a Bayesian network-based classifier and threat intelligence process with up-to-date XSS attack patterns used for detecting newly emerged XSS attacks. Kar et al. [10] developed a model to detect SQL injection attacks by modeling SQL
queries as graph of tokens using nodes to train a Support Vector Machine (SVM), then they created a dataset by collecting queries from five different web applications. Tama et al. [11] proposed a stacked ensemble WAF model in which different ensemble models including random forest, gradient boosting, and XGBoosting were used instead of machine learning algorithms as base learners and then combined the outputs with the generalized linear model (GLM).

When analyzing the above studies, we noticed some major challenges in the development of WAF models based on traditional machine learning algorithms. One of the most important challenges was the feature selection algorithm. The success of traditional machine learning algorithms depends on the extraction process, and domain knowledge becomes important in extracting and selecting the features from HTTP requests [12]. If this process yields a selection of irrelevant or unnecessary features, overfitting (high variance) or underfitting (high bias) may occur. Deep learning algorithms solve these challenges, and thus do not need a feature selection process performed during preprocessing. Deep learning algorithms can create robust and effective classification models using datasets that consist of large amounts of data without feature selection [13].

B. DEEP LEARNING METHODS

Studies that use deep learning approaches in cybersecurity have only been published since 2015, it is a new and up-to-date area of study [3]. These studies show that many deep learning methods have been successfully used for web attack detection. Most studies where deep learning methods were applied in the field of web security are dated from 2017-2020 [14].

Some CNN-based web intrusion detection systems are proposed in the literature [13], [15], [16]. In studies [13], [15], which use the CNN algorithm and inline word embedding, the features are extracted from payloads in web requests, studies [15], [16] use similar methods, however the features are extracted from the URL parts. Similarly, the authors in [17] developed a deep learning WAF model using the word2vec embedding method and Bi-LSTM architecture. In all four studies in [13], [15]–[17], the CSIC-2010 dataset is used and the detection rates are 96.13% [13], 97.07% [15], 96.49% [16] and 98.35% [17], respectively. This shows that the Bi-LSTM architecture can better represent web attacks thanks to its structure. This is explained in detail below.

Some of the studies [9], [13], [17] on web security use only the payloads, some studies [8], [16], [18] use only URL-based features. In the GET method, client payloads are sent within the URL, while the response contains the payload. In the POST method, client payloads are sent through the HTTP request along with the HTTP request body. Therefore, the POST method is preferred when sending long or sensitive inputs, such as authorization data. It is impossible to detect web attacks sent through payloads in the POST method using models created by only URL-based features; because payloads are not included in the URL. Although URL and payload features contain very important information about web attacks, attack patterns can be found in other features of the request. However, it is impossible to detect some types of attacks such as XXE, Broken Authentication or session hijacking with only these features. In cases where file upload vulnerabilities are exploited, such as XXE or shell script injection, the content and meta information of the files are needed for the analysis. Malicious files usually contain OS commands for command injection attacks, and intrusion can be detected through a detailed analysis of the file content. In session hijacking attacks, session tokens are stolen. Thus, cookie information, which is not located in the URL or the payload sections of the HTTP request, should also be taken into account. It is not possible to detect these attacks using only the URL and the payloads sent. It is possible for all parameters in web requests to contain attack payloads. In this study, all of the features in HTTP requests are used by passing the request through both word tokenization and word embedding processes. Thus, in addition to the URL and payload sections, attack vectors found in other features also contribute to the classification.

A look at the above-mentioned studies, shows that they usually deal with one [9], [10], [19] or two types [8], [18] of attacks (SQLi and/or XSS attacks). This study covers many types of attacks (SQLi, XXE, LFI, Command injection, Open Redirect, information gathering, files disclosure, CRLF (Carriage Return and Line Feed) injection, XSS, Server Side Include (SSI)). Our study stands out among the other studies because it considers multiple types of attacks, pre-processing method, developed models, and classification success.

III. PRELIMINARIES OF FEATURE CONSTRUCTION AND BI-LSTM

In this section, preliminary information about the feature construction method used, the general structure of the Bi-LSTM architecture, and its mathematical representation are presented.

A. FEATURE CONSTRUCTION

In traditional machine learning algorithms, two challenges to overcome are extracting features from the raw data and feature selection to be used in classification. Additionally, in state-of-the-art methods such as deep learning, feature extraction and feature reduction can be embedded in the models, and feature engineering operations are avoided [20]. WAF systems must look at the content of the request to decide whether the HTTP traffic contains an attack. HTTP requests consist of text-based fields such as “method”, “protocol”, “uri”, “path”, “cookie”, “content-length”, “content-type” and “body”. In neural network architectures such as LSTM, text or categorical data cannot be given as direct input to the training network; they must be converted into numerical sequences. This process includes two steps: word tokenization which creates tokens of the word vectors in the request, and word embedding which maps the text. There are different word embedding methods (Word2vec, GloVe, fastText, one
hot encoding etc.) that can be used for creating word vectors from text-based data.

In this study, sequences were created from text-based HTTP requests, using the Keras word tokenizer and the Keras word embedding functions. In the tokenization stage, categorical integer numbers are given after determining the most frequent terms by creating all possible term vectors in web requests. During the tokenization process, an efficiency analysis is carried out using a varying number of terms in order to determine how many terms will be sufficient to ensure the success of the classification. Figure 1 shows the classification success based on the varying term counts on our new dataset. It was found that the most frequent 3000 terms are a sufficient amount for learning in the tokenization process. Post-padding is processed by adding zeros to the tail of vectors to ensure the same dimensions. Thus, all possible attack patterns included in the HTTP requests could be handled regardless of the term length.

Web requests consist of text-based data. NLP methods are used in the data fusion, pre-processing, and data association processes. Web requests contain special characters as well as alphanumeric characters. When the data is being processed, sub-variations of the characters contained in the web requests are handled as word groups. In some studies [13], [15], [16], only letters are considered and the special characters in the web requests are ignored. However, attack patterns often contain special characters. In this study, special character cleaning was not performed and all characters were taken into account. Word tokens are created according to the frequency and association analysis of the sub-variations of the character groups in the web requests in the dataset. The most frequent words in the data set are determined by considering the associations of the characters and their frequencies in the whole dataset. The data association process used in this study worked effectively in detecting attack patterns that also contained special characters. However, since the bag of words technique was not used, meaningful relationships between words could not be fully addressed. Instead, features based on the statistical associations of character groups were created within the scope of the study.

B. DEEP LEARNING ALGORITHMS USED IN MEASURING THE EFFECTIVENESS OF THE PROPOSED DA-SANA

In the study, web anomaly detection models based on state-of-the-art deep learning algorithms were developed to measure and compare the effectiveness of the DA-SANA technique. Bi-LSTM, LSTM, GRU, CNN and DNN algorithms were used to develop models with and without DA-SANA. The main motivation for choosing these algorithms was that they are used extensively in the processing of texts such as web requests. GRU and LSTM are two architectures that were developed to solve the vanishing gradient problem in RNN. They are very similar and share similar successes working in areas such as sound processing [21], [22]. However, the structure of the GRU is different, the number of parameters is less than LSTM, and the GRU does not contain the output gate, unlike LSTM [22]. The structure of the LSTM is presented in detail below.

The DNN algorithm is a feed-forward neural network (FFNN). It is the most basic DL algorithm that can successfully model complex problems with nonlinear functions, thanks to the increase in the number of layers and neurons (deep concept). CNN architecture can generate robust generalization and representation vectors through convolutions and pooling processes [13]. The CNN architecture has a series of convolution layers that convolve inputs, provide feature maps and transfer them to the next layer. Although it is generally used in the field of image processing, it can also be used effectively in text processing like web requests [16].

Out of the different deep learning models we developed on our new dataset, the most successful was the Bi-LSTM. Within the scope of this study, the Bi-LSTM model reached the highest average classification success on the ECML-PKDD and on our new dataset in multi-class web anomaly detection.

For this reason, a WAF model based on Bi-LSTM was proposed within the scope of the study. The general structure of the Bi-LSTM architecture and the proposed model developed based on Bi-LSTM are presented in detail below. Model architecture based on Bi-LSTM used in the study is shown in Figure 2

C. BIDIRECTIONAL LONG SHORT TERM MEMORY (Bi-LSTM)

In deep learning architectures, computational operations performed in the single-layer structure of traditional machine learning algorithms were calculated gradually through layers of different abstraction levels and multiple parameters. Outputs calculated in each layer are transferred to other layers and then extract representation using the most significant features of the data with non-linear functions. This study proposes, Bi-LSTM based binary-class and multi-class classification deep learning models. The basic properties
and mathematical formations of the Bi-LSTM architecture are presented below. Recurrent Neural Network (RNN) is successful in predicting the next sequence in short term dependencies. However, it does not succeed in long-term dependencies problems due to gradient vanishing or exploding [21]. LSTM [22] architecture is a special version of RNN architecture. Due to its long and short term memory, it solves the problem of long-term connections that exist in RNN architectures [22]. In this LSTM architecture, the structure is the same as RNN architecture, but the neurons vary. These neurons allow for long term connections to be kept while updating the cell status with three gates, input, output, and forget [22]. The Bi-LSTM architecture [23] was created by connecting the LSTM recurrent layer in opposite directions of each other in order to produce a single output (see Figure 2).

In a standard RNN architecture, when \( \mathbf{x} = (x_1, x_2, x_3, \ldots, x_n) \) is given the input sequence, the hidden vector array \( \mathbf{h} = (h_1, h_2, h_3, \ldots, h_n) \), and output vector array \( \mathbf{y} = (y_1, y_2, y_3, \ldots, y_n) \) is calculated in \( n \) steps as follows, from \( t = 1 \) to \( t = n \) (see Eq. 1-2) [21]. Here \( \mathbf{W} \) denote weight matrices, \( \mathbf{b} \) denotes bias values, and \( \mathbf{H} \) denotes the hidden layer function (generally sigmoid).

\[
h_t = H(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (1)
\]

\[
y_t = W_{yh}h_t + by \quad (2)
\]

The hidden layer function \( H \) in the LSTM architecture, which was developed to keep long-term connections and consists of input, output and forget gates, is calculated as in Eq. 3 – 7 [21].

\[
f_t = \sigma_a (W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (3)
\]

\[
i_t = \sigma_a (W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (4)
\]

\[
o_t = \sigma_a (W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \quad (5)
\]

\[
c_t = f_t c_{t-1} + i_t \sigma (W_{cx}x_t + U_c h_{t-1} + b_c) \quad (6)
\]

\[
h_t = o_t \sigma_h (c_t) \quad (7)
\]

In Bi-LSTM architecture, the outputs of the forward cells (as in Eq. 8) and backward cells (as in Eq. 9) are combined with a single activation function and the output sequence \( \mathbf{y} \) (as in Eq. 10) is calculated as follows [23].

\[
h_f = H(W_{xf}x_t + W_{hf}h_{t-1} + b_h) \quad (8)
\]

\[
h_b = H(W_{xb}x_t + W_{hb}h_{t-1} + b_h) \quad (9)
\]

\[
y_t = (W_{yh}h_f + W_{yh}h_b + b_y) \quad (10)
\]

**IV. METHODOLOGY**

In this section, the DA-SANA method developed in the study and the theoretical representation of the proposed classification models are presented. The proposed data augmentation
method and classification models are discussed in detail using structural terms. Within the scope of the study, the DA-SANA method based on the total character length included in the web requests was developed, while the sensitivity of the model was increased, the FP rate, which is an essential problem in web attack detection, was reduced with the goal to prevent overfitting. With the DA-SANA process detailed below, the original data in the learning set was concatenated with noisy data; thus, the amount of data was double augmented.

A. DA-SANA

Since web requests are very similar, determining web attack traffic with high sensitivity via present NLP methods has become an important problem. Increasing the model sensitivity by adding noisy data is one method used to combat this problem and is recognized in the literature. In this study we developed the DA-SANA method, which adds self-adapted noise based on the length of the request. This makes it an effective method for distinguishing normal and abnormal requests. To the best of our knowledge, this method has never been used for intrusion detection systems. The statistical analysis performed in this study and some other studies [5], [24], [25] clearly revealed that the length of the HTTP request is significant in web attack detection. Therefore, we chose to use lengths of web requests as self-adaption coefficients when adding noise.

1) THE SIGNIFICANCE OF WEB REQUEST LENGTH IN CLASSIFICATION

In one of the first studies in the field of anomaly-based web application [5], the features included in the web requests were handled in order to separate the normal traffic from the abnormal. According to this study, HTTP requests to a web application have a similar distribution of lengths. Web attack requests can generate web requests that are longer or shorter than the web application would normally receive. This causes big changes in the variance on the distributions of web request lengths.

In this study, the extra trees classifier based feature selection process was performed on different web anomaly datasets such as CSIC-2010 [26], ECML/PKDD [27] and our new dataset. The extra trees classifier creates de-correlated decision trees based on the Gini index. It is an ensemble algorithm, very similar to the random forest algorithm. Gini indexes based on the extra trees can be used as significances of a feature set for feature selection. The most important five features, based on the extra tree classifier for three different web anomaly datasets, are shown in Figure 3. In addition, the names of these features are shown in Table 1, since some features were created as a result of tokenization, the feature names were determined by analyzing a tokenized word list of the web requests.

Consequently, the length of the HTTP request was the most significant feature for CSIC-2010 and our dataset, and the second most significant feature for ECML/PKDD dataset, contributing 25.16%, 6.33% and 2.41%, respectively. The noise coefficient denotes the density of the noise when creating a new noisy request from the original request. It was decided to calculate the noise coefficients, used in creating the self-adapted noisy data, in direct proportion to the length of the web requests. The effectiveness of the DA-SANA method is seen in the rest of the study in both the results and comparison sections.

2) THEORETICALLY BASICS OF DA-SANA

The pseudocode of the DA-SANA noise adding algorithm is shown as Algorithm I. The definitions of the symbols used in the equations and Algorithm I are shown in Table 2. The creation of noisy data with DA-SANA was carried out in two stages. Firstly, the density and locations of the noisy data were randomly determined based on $C_i$ coefficients specified against the length of the requests. Secondly, $N_i$ noise dataset
TABLE 1. Top five most significant feature names based on extra tree-based feature selection.

| Dataset       | 1st significant | 2nd significant | 3rd significant | 4th significant | 5th significant |
|---------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| CSIC-2010     | Request-Length  | Path            | Payload-Key-First | Content-Length  | Payload-Value-First |
| ECML/PKDD     | inContext       | Request-Length  | Payload-Key-First | Header          | Payload-Value-First |
| OUR-DATASET   | Request-Length  | URI-End         | URI-Middle       | Payload-Value-First | User-Agent       |

was created in dataset sizes and contained entirely randomly determined tokens \((k \subseteq (f, 2*f))\) in the padding section; which were not included in the dataset. The \(f\) term denotes the most frequent terms count (set 3000 in the study) in the tokenization process. The amount of noise to be added to a request was calculated according to the noise coefficients calculated as Eq. 12. Here, \(C_i\) represents the noise coefficients, \(L\) is the number of items included by HTTP requests, and \(n\) is the count of the features in the sequences after post-padding.

The binomial distribution was used (see in Eq. 11), according to the discrete probability distribution (0 or 1), to specify the locations of the noisy values in the sequences.

The Gaussian normal distribution (as shown in Eq. 13) was used to determine the noise tokens, where parameter \(\sigma\) denotes the standard deviation, and \(\mu\) denotes the mean or expectation of the distribution. \(N_i\) dataset containing noisy data was created with the \(S_i\) original sequences depending on the \(C_i\) noise coefficients and Gaussian distribution (as in Eq. 14). Finally, data augmentation was realized by adding \(N_i\) noisy dataset to \(S_i\) original dataset.

\[
P_{\text{Binomial}}(x) = \binom{n}{x} p^x q^{n-x} = \frac{n!}{(n-x)!x!} p^x q^{n-x} \quad (11)
\]

\[
C_i = P_{\text{Binomial}} \times \left( \frac{L_i - \min(L)}{\max(L) - \min(L)} \right) \ast 0.05, \text{size} = (n) \quad (12)
\]

\[
P_{\text{Gaussian}}(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{x-\mu}{\sigma} \right)^2} \quad (13)
\]

\[
N_i = S_i + P_{\text{Gaussian}} (k \leq (f, 2*f), \text{size} = n) \ast C_i \quad (14)
\]

The amount of noise to be added was determined in direct proportion to the length of each request, and was limited to a maximum of \(20\%\) of the sequence length \((n)\). The limitation rate was specified by an analysis performed on our dataset. The detection rates of the multi-class models based on Bi-LSTM with Da-SANA at varying limit rates, are presented in Figure 4. The purpose of applying the limit rate was to restrict the change in the frequency of tokens with sequences, thus preventing distortion of the representations in the dataset. Figure 4 shows that the highest detection rate was achieved when the number of noise terms added to the sequence, by the proposed DA-SANA technique, was limited to 20% of the request length. While the low amount of noisy terms will eliminate the effectiveness of DA-SANA, the high number of noisy terms may cause the augmented web request to diverge from the original.

In the DA-SANA method, tokens different from the existing tokens in the sequence are added as noise; the frequency of the existing tokens does not change. Additionally, since the original dataset is included with the augmented dataset, it ensures that the models can see the original web requests in the training and testing processes. The amounts of noise were also limited and there was no change in the label distribution of the web requests (as seen in Tables 4 and 5). Thus, the proposed DA-SANA method caused a negligible change representation of datasets.

TABLE 2. Definitions of symbols in equations and algorithm.

| Symbol | Definitions |
|--------|-------------|
| \(x\)  | input vector |
| \(C_i\) | noise coefficients based on length of requests |
| \(N_i\) | noisy dataset |
| \(L_i\) | lengths (counts of characters) of HTTP requests |
| \(S_i\) | original dataset (sequences after tokenization) |
| \(m\)  | number of HTTP requests in the input set |
| \(n\)  | total number of features after post-padding maximum |
| \(f\)  | frequent terms count (set 3000 in study) |

Algorithm 1 DA-SANA Algorithm

Input: Training dataset \(\Omega \leftarrow \{s_1, s_2, \ldots, s_m\}\) with HTTP Request \(x_i\),

Sequences \(S \leftarrow \{(s_{11}, s_{12}, \ldots, s_{1n}), \ldots, (s_{m1}, s_{m2}, \ldots, s_{mn})\}\) with tokenized sequence item \(s\)

Output: Augmented training dataset

1. for \(\forall x_i \in \Omega\) do
2. \(\text{Set: } L_i \leftarrow \text{length of } x_i \text{ // character count of } x_i\)
3. \(\text{Set: } L_i \leftarrow \frac{L_i - \min(L)}{\max(L) - \min(L)} \text{ // normalize between (0, 1)}\)
4. \(\text{Set: Noisy Data } N_i \leftarrow \text{random generated } n \text{ integer between } (f, 2*f) \text{ with a Gaussian normal distribution} \)
5. \(\text{Set: Noisy coefficients } C_i \leftarrow \text{random generated } n \text{ integer between (0, 1) and up to } 20\% \text{ of } n \text{ may contains ones.} \)
6. \(\text{Set: Noisy dataset } ND_i \leftarrow N_i \ast C_i + S_i \ast C_i' \)
7. end for
8. \(\text{Set: Augmented training dataset } S' \leftarrow S_i + ND_i \)
B. HTTP REQUESTS ANOMALY DETECTION MODELS

Within the scope of the study, both binary-class and multi-class classification models were developed. In order to understand and analyze the processes of anomaly-based WAF systems, we first present the theoretical structure of the WAF systems.

1) THE THEORETICAL BACKGROUND OF WAF SYSTEMS

An anomaly-based WAF is represented by a set of six items \( (D, Y, X, P, H, L) \), where the first three items are related to data structures, and the remaining three items are related to the prediction model. Anomaly-based WAF systems basically consist of the processing and the classification of the data sources containing web requests. Theoretical representations of the six formal items that create an anomaly-based WAF system are presented below.

**D**: the data source that provides the data for the WAF to be trained, tested, and analyzed. Fundamentally, this is a stream of web requests that consist of consecutive data items. Each WAF system has its own data representation. While some WAFs only use the web requests or certain parts of them (URL, Payload, Method, Body, etc.), some WAFs may also include the web server’s responses and the response codes. In general, it is defined as \( D = (D_1, \ldots, D_N) \) where \( D_i \) is the \( i^{th} \) web request data, that will be analyzed by the WAF, which consists of web request parts.

**Y**: a set of labels define as \( Y = (Y_1, \ldots, Y_N) \) where \( Y_i \) indicates whether the \( D_i \) web request is normal or abnormal (or which type of attack). Labels in the \( Y \) set are pre-tagged and are all considered true. For binary-class web anomaly detection, \( Y_i \) is the \( i^{th} \) item in \( Y \) set, and \( Y_i \in \{ \text{Normal, Abnormal} \} \) or \( Y_i \in \{0,1\} \) where 0 denotes normal, 1 denotes abnormal. For multi-class classification, we can create the \( Y \) set with \( Y_i \in \{ \text{Normal, Attack-Type-1}, \ldots, \text{Attack-Type-}K \} \) or \( \{0, 1, \ldots, K\} \) where \( K \) is the number of attack types.

**X**: a feature vector contains a finite number of features, formally \( X = [X_1, \ldots, X_m] \) where \( m \) is the number of features.

Features can be constructed with different methods such as Ngram, Tf-Idf, or word embedding for each WAF.

**P**: data preprocessing stages contain feature selection, data reduction and representation. Feature selection and data reduction operations are optional and may not be needed for some algorithms such as deep learning techniques. However, textual web requests need to be converted to representation in the \( X \) feature vector space, where \( P: D \rightarrow X \), that the classification algorithm can handle.

**H**: classification algorithm. Actually, a function that weights the input features so that the output separates one class into normal and abnormal (or attack types). According to the problems, different machine learning algorithms can be selected as classifiers or hybrid structures can be created in which different algorithms are used together as an ensemble model.

**L**: classifier model that maps feature vector representation \( X \) of given data source \( D \) to labels in elements of set \( Y \).

In consideration of the formal definitions given above, mathematical representation of HTTP request traffic classification process is shown with the following Eq. (15-17).

**Dataset Pre-processing and Splitting:**

\[
(D_{\text{train}}, D_{\text{test}}) : P(D) \cup Y \rightarrow X \cup Y = \{x_n, y_n\}, \quad n = 1, \ldots N
\] (15)

**Classifier:**

\[
L : D_{\text{train}} \rightarrow H(X), \quad \text{(16)}
\]

where

\[
H (X_i) : X_i \rightarrow Y_i
\]

**Prediction:**

\[
Y_{\text{test}} : X_{\text{test}} \rightarrow L, \quad \text{(17)}
\]

where

\[
X_{\text{test}} = \{x_k\}, \quad k = 1, \ldots K
\]

In these equations, \( D_{\text{train}} \) denotes training data, \( L \) denotes the classifier model, \( H(X_i) \) stands for classification process based on a specific algorithm, \( X_i \) denotes the \( i^{th} \) input, and \( Y_i \) denotes the model output of the \( i^{th} \) input. \( Y_{\text{test}} \) labels are predicted using the trained \( L \) classifier model for \( X_{\text{test}} \) test-set which contains \( K \) number of unknown web requests (Eq. 17).

2) THE STRUCTURE OF THE PROPOSED MODEL

The proposed classification model consists of one embedding layer, two consecutive Bi-LSTM layers, and one softmax classifier layer, respectively (see Figure 2). For the binary-class and multi-class models, two different models with the same structure are proposed, and the details are shown in Table 3. While an embedding dimension of 16 was determined to be sufficient for successful learning in the binary-class classifier model, it was determined that 32 was needed for the multi-class classifier. The flow chart of the proposed model is shown in Figure 5. As seen in Figure 5,
the development of the web anomaly classification model was carried out in two stages. In the first stage, web requests were preprocessed, word tokens were extracted and data augmentation based on DA-SANA was performed. In the second stage, the WAF model was created by performing training and testing with ten-fold cross validation based on the deep learning model.

3) HYPER-PARAMETER TUNING
After training with the augmented training set, hyper-parameter tuning was performed to improve the classification performance of the models. Different parameter optimizations were performed to improve the performance of the models during this hyper-parameter tuning process. First, different non-linear activation functions such as Relu, sigmoid, tangent hyperbolic (tanh) were tried in Bi-LSTM layers. Results showed that the tanh function was the most successful.

Softmax was used in the output layer because it was observed as the most successful activation function as the classifier. In order to prevent overfitting in the Bi-LSTM layers, a dropout was applied at the inputs of the layers at a rate of 5% and a recurrent dropout at a rate of 5%. Due to the amount of dataset and memory constraints, batch size was set to 256. The order of the inputs in the training set was shuffled in each epoch during the training process. Otherwise, data in mini-batches will be highly correlated, causing overfitting of the model [28]. In the optimization of the model, the Adam optimization algorithm was used with a 0.001 learning rate. Binary cross-entropy and categorical cross-entropy functions were used as loss function for the binary-class and the multi-class models, respectively.

V. DATASETS AND VALIDATION METHODS
There are not too many up-to-date datasets available to test WAFs. Currently, the DARPA KDD 99 data set is the most well-known and most used dataset for testing Intrusion Detection Systems (IDS) [29]. However, this dataset is criticized because it is outdated and does not contain current attack types [30]. Within the scope of this study, a novel web attack dataset, labelled with multi attack types containing new up-to-date attack parameters, was created. In addition to our new dataset, two benchmark web anomaly datasets, that are frequently used in the field of web security CSIC-2010 and ECML/PKDD, were used in the training and testing of models. These datasets are preferred because they are directly related to web security, contain web attacks, and are publicly available.

A. CREATING A NEW DATASET FOR WEB ATTACK CLASSIFICATION
There are very few public datasets in the field of web security, and this lack of up-to-date datasets is one of the most important problems in the field. Two of the benchmark and most well-known datasets in the web security classification

| Table 3. Architectural details of proposed models. |
|--------------------------------------------------|
| **Binary-Class Classifier Model**                |
| Layer (Type) | Input Shape | Output Shape | Activation Function |
| Embedding    | (m,n)       | (m,n,16)     | -                   |
| Bi-LSTM      | (m,n,16)    | (m,n,256)    | tanh                |
| Bi-LSTM      | (m,n,256)   | (m,2)        | softmax             |

| **Multi-Class Classifier Model**                  |
|--------------------------------------------------|
| Layer (Type) | Input Shape | Output Shape | Activation Function |
| Embedding    | (m,n)       | (m,n,32)     | -                   |
| Bi-LSTM      | (m,n,32)    | (m,n,256)    | tanh                |
| Bi-LSTM      | (m,n,256)   | (m,2)        | softmax             |
| Dense        | (m,256)     | (m,label_count) | softmax             |

FIGURE 5. Flow chart of the proposed model.
field are the ECML/PKDD and the CSIC-2010, which were established creating WAF models. Therefore, we created a robust, consistent, and up-to-date multi-labelled web attack classification dataset by considering the web vulnerabilities in Owasp Top Ten [2]. Robustness is a metric that is generally used to denote the ability of the classifier models to withstand operations such as shifting data or adding noise in the dataset. In this study, the meaning of robustness refers to the datasets suitability for use with different algorithms, possession of a standard suitable for its nature, and suitable for augmentation with similar data sources, is portable, and its creation schema and sources are transparent [31]. In addition to the successful models created using our dataset confirming its compatibility with different deep learning algorithms; additional models confirmed its compatibility with other traditional machine learning algorithms. Moreover, our dataset was created following the nature of web security, and the creation method is clearly presented within the scope of the study. Our dataset can be moved to different platforms through pre-processing and is suitable for augmentation with data from similar sources. Therefore, our dataset created within the scope of this study contains all the above robustness features.

We prepared two real websites to create the dataset. One of them was an e-commerce application, the other was a general web application based on WordPress, the most used Content Management System (CMS) in the world. First, all HTTP requests and parameters that websites could receive, were scanned and recorded using spider tools (Paros and Spider). Then, the payloads used in the realization of the attacks were collected by scanning all of the resources we could access on the internet, in great detail. Especially, GitHub and open source web attack tools, both pre-processed and combined. Similarly, the payloads used in the creation of normal HTTP requests were chosen from the most commonly used words in the world in accordance (e.g. name, surname, city, country etc.) with the parameters of HTML requests. HTTP requests created using normal payloads were recorded in the database with the Valid label, using a network sniffer tool on the web servers. Similarly, HTTP requests created with attack payloads were recorded in separate databases according to the attack types. While creating the attack traffic, attention was paid to make sure the attacks types were appropriate to the parameters included in the requests, and no attack type was carried out using all the pages or parameters. The Burp Suite tool was used to generate the web attacks and the TcpFlow was used for sniffing. The topology used to create the dataset for web attack classification is shown in Figure 6.

The labels and distributions of the HTTP traffic in our dataset are shown in Table 4. There are nine different labels in the dataset, one is “Valid” and the rest are attack types. The dataset contains 108545 HTTP requests in total, which is distributed as 73529 “Valid” and 35016 attack requests. The distribution of normal and abnormal data in our dataset is 67.74% and 32.26%, respectively. Due to the nature of the intrusion detection, our dataset is imbalanced. To the best of our knowledge, at the time of this study, there was not a comprehensive public dataset in the literature.

B. DESCRIPTIONS OF OTHER DATASETS USED IN STUDY

1) CSIC-2010 HTTP DATASET

The CSIC-2010 dataset was produced by the National Research Council of Spain (CSIC), the Information Security Institute, by recording both normal and attack traffic for a web application [26]. The data set contained 61065 HTTP data in total, which was distributed as 36000 normal and 25065 attack requests. These HTTP requests were labelled as either normal or anomaly, and the dataset is only suitable for binary-class classification. Using the features, they contained in the pre-processing stage, the HTTP requests were divided into different columns: Method, URI, Path Payload Keys, Payload Values, etc.

2) ECML PKDD DATASET

This dataset was published at the ECML/PKDD conference in 2007, as a part of the Analyzing Web Traffic ECML/PKDD 2007 Discovery Challenge [27]. HTTP requests consist of 6 different components; method, protocol, uri, query, headers and body. There are 52296 HTTP requests in 8 different classes, including “Valid” and seven different types of web attacks. The dataset is imbalanced because more than 66%
of the data are labelled as “Valid”. The distribution of the labels in the dataset and the situation of the training set before and after the data augmentation with DA-SANA is shown in Table 5.

C. EXPERIMENTAL TOOLS

Kali Linux, Burp Suite, Paros and TcpFlow were used as web spider, web attacking and network sniffer tools while creating the web anomaly dataset. Keras and Tensorflow open source platforms and Python programming language were used in the execution of the study. Additionally, panda, numpy, scikit learn, matplot libraries were used for the data pre-processing, statistical analysis and visualization. In the training and testing of the models, a computer with Intel 6850K 3.6 GHz i7 Processor, 16 GB RAM, GTX 1080 Ti 11 GB GPU, 512 GB SSD configuration was used.

TABLE 5. Distribution of ECML PKDD dataset.

| Type            | Size  | After Data Augmentation | Dist. |
|-----------------|-------|-------------------------|-------|
| Valid           | 35006 | 70012                   | 66.94%|
| Ldap Injection  | 2279  | 4558                    | 4.36% |
| OS Commanding   | 3420  | 6840                    | 6.54% |
| Path Traversal | 3029  | 6058                    | 5.79% |
| SSI             | 1898  | 3796                    | 3.63% |
| SQLi            | 2560  | 5120                    | 4.90% |
| XPath Injection | 2279  | 4558                    | 4.36% |
| XSS             | 1825  | 3650                    | 3.49% |
| Total           | 52296 | 104592                  | 100%  |

D. VALIDATION METHODS

In this study, five different models were developed for web attack detection: three models for binary-class detection based-on the CSIC-HTTP, ECML-PKDD and our new dataset, and two models for multi-class detection based-on the ECML-PKDD and our new dataset. Two well-known data splitting methods were used to validate the developed models and measure their performances. One of these methods was to split the dataset into three sets with a percentage split as training, validation and test sets. The datasets were divided into random sets of 60%, 20% and 20%, respectively, as training, validation, and testing. The second method was the k-fold cross-validation, especially in 10-fold, which allowed for the whole dataset to be used in training and testing. In k-fold cross-validation, the dataset was divided into k parts randomly, the k-1 parts were used in training, and the remaining one part was used in testing. This process was completed by repeating k times so that the whole dataset could be used in the training and testing stages.

E. PERFORMANCE METRICS

In this section, the explanations of the metrics used in performing the performance measurements of the models are briefly given. The imbalanced dataset caused a classification problem that is prone to bias due to skew in the class distribution in the dataset [32]. Classifying imbalanced datasets is a major challenge, as machine learning algorithms using classification generally work well for homogeneous structures with the same number of labels. Typically, web anomaly detection is an imbalanced classification problem where the majority class is normal and the minority class is abnormal. In imbalanced classification problems, since the distribution of classes is skewed, it is not sufficient to use the accuracy or error rate alone as a metric for model performance. In addition, it is important to calculate the Precision, Recall, F-measure and False Alarm Rate (FAR) metrics, which take into account the sensitivity of minority classes [33]. The performance metrics used in the study are presented in Table 6, where the TP parameter denotes True Positive, TN denotes True Negative, FP denotes False Positive, FN denotes False Negative. In the evaluation of the performance metrics, weighted average (W. Avg.) values, which are based on the distribution of class, are used, since they give more accurate results on multi-class classification.

TABLE 6. Performance metrics.

| Metrics       | Explanations          |
|---------------|-----------------------|
| Accuracy (Acc)| (TP + FP)/(TP + FP + TN + FN) |
| Precision     | TP/(TP + FP)          |
| Recall        | TP/(TP + FN)          |
| F1-Score      | 2* Precision * Recall / Precision + Recall |
| FAR           | FP/(FP + TN)          |

VI. RESULTS AND DISCUSSIONS

In this section, the performance results of the models for three different datasets are presented below.

A. PERFORMANCE EVALUATIONS

1) PERCENTAGE SPLIT VALIDATION

The training and testing processes, for each model, were repeated three times experimentally; with percentage split for three datasets. The confusion matrices and performance evaluations for the binary-class classification models with percentage split are shown in Table 7. According to the results of the three experiments of the binary-class classification model, in the least, the classification successes for our dataset, ECML/PKDD and CSIC-2010 are 98.34%, 97.0% and 99.08%, respectively.

Considering the precision, recall and f1-score values of the models, it can be seen that the sensitivities of the models are very high, the FN and FP ratios are low, and in the least, the F1-scores for our dataset, ECML/PKDD and CSIC-HTTP are 98.34%, 96.95% and 99.06% respectively.

Multi-class classification can be performed for the ECML/PKDD and our dataset since they have multi-class labels. The confusion matrices and performance evaluations for the multi-class classification models of these datasets
TABLE 7. Confusion matrices and performance evaluations of binary-class classification based-on Bi-LSTM with percentage split.

| Actual Class | Predicted Class | Performance Evaluations |
|--------------|----------------|-------------------------|
| OUR DATASET  |                |                         |
| Normal       | 29245          | 0.9815                  |
| Abnormal     | 552            | 0.9877                  |
| Total / W. Avg | 29797        | 0.9834                  |
| ECML/PKDD    |                |                         |
| Normal       | 6994           | 0.9572                  |
| Abnormal     | 312            | 0.9996                  |
| Total / W. Avg | 7306        | 0.9712                  |
| CSIC-2010    |                |                         |
| Normal       | 7169           | 0.9927                  |
| Abnormal     | 52             | 0.9879                  |
| Total / W. Avg | 7221        | 0.9907                  |

based on percentage split are shown in Table 8, together. As a result of 3 experiments of the multi-class models, in the least, 41922 of 43418 HTTP requests in the test set for our dataset and 9824 of 10460 for ECML/PKDD are correctly classified and the classification successes are 96.5% and 93.91%, respectively. When we look at the Table 8, for our dataset, it shows that the best classification rate was achieved for the Valid class with a 98.6% F1-score, the Open Redirect attack follows with a 96.7% score, and the worst F1-score belongs to the Command Injection attack with 78.2%. Since arbitrary commands are contained in the command injection attack, the model is confused by the XSS, SQLi and LFI attack patterns that contain similar commands.

Additionally, when looking at the results of the ECML/PKDD, the best classification rate was achieved for the XPath injection attack and all attacks were detected; no false negative (FN) was produced for this type. The F1-score of the Valid, Ldap injection and XSS types, were above 0.99 and had the best values after Xpath Injection, respectively. The worst classification performance occurred in the SSI attack type for ECML/PKDD, and as the confusion matrix in Table 8 shows, the types of attacks that caused confusion

TABLE 8. Confusion matrices and performance evaluations of our dataset and ECML/PKDD by multi-class based-on Bi-LSTM with percentage split.

| Actual Class | Predicted Class | Performance Evaluations |
|--------------|----------------|-------------------------|
| OUR DATASET  |                |                         |
| CRLF         | 957            | 0.975                   |
| Command Injection | 5     | 0.80                   |
| LFI          | 1015           | 0.780                   |
| Open Redirect | 70         | 0.784                   |
| SQLi         | 11             | 0.995                   |
| Valid        | 12             | 0.954                   |
| XSS          | 1              | 0.920                   |
| XKE          | 1              | 0.965                   |
| Total / W. Avg | 981       | 0.965                   |
| ECML/PKDD    |                |                         |
| Ldap Injection | 466       | 0.989                   |
| OS Commanding | 587         | 0.739                   |
| Path Transversal | 0    | 0.787                   |
| XSS          | 3              | 0.939                   |
| SQLi         | 14             | 0.858                   |
| Valid        | 5              | 0.891                   |
| XPath Injection | 0        | 0.932                   |
| XSS          | 0              | 0.939                   |
| Total / W. Avg | 471       | 0.932                   |
with the SSI are the OS Commanding, Path traversal and SQLi, respectively. This is because an SSI attack contains script words and file paths, like the above attack types, as it allows the script to be injected through HTML pages or to run common commands. However, even if the types of these attacks were misclassified, the classification of the attacks as normal appears to be very low in multi-class classification.

2) 10-FOLD CROSS VALIDATION
All models were trained with 10-fold cross-validation and the performance tests were carried out. Performance evaluations based on the 10-fold cross-validation are presented in Table 9. Considering the results in terms of binary-class models based 10-fold cross validation, the best detection rate belongs to ECML/PKDD with 99.5%, the lowest detection rate belongs to our dataset with 97.5%, and the detection rate of the CSIC-2010 model is 98.6%.

The performances of the multi-class models trained by applying DA-SANA compared to without DA-SANA are shown together in Table 9 for comparability. As shown in Table 9, while the same model had a 96.6% accuracy for multi-class classification with DA-SANA, it could only have an 89.7% accuracy without DA-SANA for our dataset based on 10-fold cross validation. Similarly, the classification accuracies were 92.7% and 87.7% for ECML / PKDD based models with and without DA-SANA, respectively. So, with DA-SANA, 6.8% improvement was achieved for our dataset and 6.2% for ECML/PKDD. Therewithal, it can be clearly seen in Table 9 that the DA-SANA method had a significant contribution in terms of accuracy, precision, recall and F1-score performance metrics. The effectiveness of data augmentation based on the DA-SANA method on the imbalanced web anomaly dataset is explicitly shown in Table 9.

Class-based performance evaluations of the Bi-LSTM models with DA-SANA based on 10-Fold cross-validation for our dataset (a) and ECML/PKDD (b) are shown Figure 7 in detail. As seen in Figure 7 (a), the best classification performance measurements according to F1-score for our dataset were achieved in SSI, SQLi and CRLF classes as 98.4%, 96.5% and 93.9%, respectively. However, Command

**TABLE 9. Performance evaluations of all models based on Bi-LSTM with 10-fold cross validation.**

|                  | OUR DATASET                      | ECML-PKDD                      | CSIC-2010                      |
|------------------|----------------------------------|--------------------------------|--------------------------------|
|                  | Binary-Class                     | Multi-Class                    | Binary-Class                   | Multi-Class                   | Binary-Class                     |
|                  | With Da-SANA                     | Without Da-SANA                | With Da-SANA                   | Without Da-SANA               | With Da-SANA                     |
| Precision        | 97.6%                            | 88.6%                          | 96.6%                          | 99.4%                          | 86.1%                            | 92.6%                            | 98.2%                          |
| Recall           | 97.5%                            | 89.7%                          | 96.5%                          | 99.8%                          | 87.7%                            | 92.7%                            | 98.4%                          |
| F1 Score         | 97.5%                            | 88.3%                          | 96.5%                          | 99.6%                          | 85.8%                            | 92.3%                            | 98.1%                          |
| Accuracy         | 97.5%                            | 89.7%                          | 96.6%                          | 99.5%                          | 87.7%                            | 92.7%                            | 98.6%                          |

**FIGURE 7.** Class-based performance evaluations of our dataset (a) and ECML/PKDD (b) based on Bi-LSTM with 10-fold cross-validation.
injection attack type had the lowest F1-score with 81.1%, similarly as percentage split (see Table 8). It seems that the Command injection attack was mostly confused with LFI, SQL, and XSS attacks (see Table 8). This is because, these attack types contain similar command words, and the dataset was imbalancedly distributed. However, when compared to the model without DA-SANA, it was observed that a significant improvement was achieved in the attack type with data augmentation based on the proposed DA-SANA method, especially in attack types with limited number data. Furthermore, while the best classification success, according to F1-score on label based ECML/PKDD was achieved in the Valid, XPath and Ldap injection classes as 99.8%, 94.5% and 92.9%, respectively (see Figure 7). Nevertheless, SSI attack type had the lowest F1-score with 39.2%, similarly as percentage split. The reasons for low success with some labels, like SSI, in multi-class classification are the imbalanced distribution of the ECML/PKDD (see Table 5), and having attack payloads that are very similar to other attacks; as explained above. To solve the imbalanced data problem, data shuffling and stratified sampling were applied in the cross-validation process. It is seen that the percentage split and cross-validation results fit each other, and there is no significant difference (see Table 8 and Figure 7).

### B. THE EFFECTIVENESS OF DA-SANA IN WEB ATTACK DETECTION

To analyze the improvement that the DA-SANA provided in the classification of web attacks, the model results are comparatively analyzed in this section. To show the effectiveness of the developed DA-SANA technique on deep learning algorithms, models based on LSTM, GRU, CNN and DNN algorithms were developed in addition to Bi-LSTM. Comparisons of the classification accuracies of the state-of-the-art DL models with and without DA-SANA are presented in Table 10. The detection rates of the proposed models based on Bi-LSTM, that were used in Table 10, consist of the highest values of the 10-fold cross-validation and the percentage split. The remaining four DL models were validated by the percentage-split method. It can be seen that the DA-SANA provides improvements for both two-class and multi-class web anomaly detection in all models, based on the deep learning algorithms in the study. The lowest average improvement rate for two-class detection was in the DNN model, the highest was reached in the CNN model, with an improvement of 0.62% and 2.9%, respectively. While the lowest average improvement rate with DA-SANA for multi-class detection was in the CNN model, the highest was reached at the LSTM model, with an improvement of 4.52% and 7.77%, respectively. According to the average improvements of all the DL models on the three datasets, it is seen that it provided improvements for binary-class and multi-class, 1.61% and 6.52% respectively while using DA-SANA. It is noticed that DA-SANA cannot provide a very high improvement in the already high two-class classification performances. However, a high improvement can be seen for multi-class web anomaly detection with DA-SANA. The most important reason for this improvement is that the amount of noise to be added in the newly created noisy request sequence was determined based on the length of the web requests. The lengths of the web anomaly requests caused a significant change in the distribution of request lengths expected by the web application. With DA-SANA, the variance of web request lengths can be transferred in direct proportion to the classification model as the amount of noise. Consequently, high sensitivity can be achieved against web attack requests with DA-SANA.

The data shows that the DA-SANA provided an average improvement of 0.6% and 6.54% for the proposed Bi-LSTM model in two-class and multi-class classification, respectively. While the proposed Bi-LSTM model had the highest detection rate for our new dataset in multi-class web anomaly detection, it reached the second-highest detection rate for ECML-PKDD. Bi-LSTM works better in sequence classification problems thanks to its bidirectional structure [23]. Thus, Bi-LSTM can create models with better representation for web anomaly detection by creating strong connections.

### TABLE 10. Accuracy comparisons of state-of-the-art DL models with and without DA-SANA.

| Model | Dataset | BINARY-CLASS | MULTICLASS |
|-------|---------|--------------|------------|
|       | without | with DA-SANA | without | with DA-SANA |
| Bi-LSTM | OUR DATASET | 97.65% | 98.34% | 89.76% | 96.59% |
|        | ECML/PKDD | 98.94% | 99.47% | 87.66% | 93.91% |
|        | CSIC-2010 | 98.35% | 99.08% | N/A | N/A |
| Average | 98.31% | 98.96% | 88.71% | 95.25% |
| Avg. Imp. | 0.65% | 6.54% |
| LSTM | OUR DATASET | 97.58% | 98.19% | 87.47% | 95.20% |
| | ECML/PKDD | 98.54% | 99.37% | 81.52% | 92.22% |
| | CSIC-2010 | 92.87% | 95.83% | N/A | N/A |
| Average | 96.33% | 97.79% | 84.69% | 92.46% |
| Avg. Imp. | 1.46% | 7.77% |
| GRU | OUR DATASET | 97.83% | 98.34% | 88.25% | 92.51% |
| | ECML/PKDD | 98.64% | 99.56 | 82.81% | 91.87% |
| | CSIC-2010 | 93.27% | 96.23% | N/A | N/A |
| Average | 96.58% | 98.04 | 85.53 | 92.19 |
| Avg. Imp. | 1.46% | 6.66 |
| CNN | OUR DATASET | 97.47% | 98.26% | 85.44% | 91.31% |
| | ECML/PKDD | 95.78% | 97.71% | 91.06% | 94.24% |
| | CSIC-2010 | 92.98% | 93.95% | N/A | N/A |
| Average | 93.74% | 96.64% | 88.25% | 92.77% |
| Avg. Imp. | 2.9% | 4.52% |
| DNN | OUR DATASET | 98.06% | 98.79% | 90.22% | 93.00% |
| | ECML/PKDD | 99.75% | 99.95% | 82.42% | 93.87% |
| | CSIC-2010 | 95.07% | 96.00% | N/A | N/A |
| Average | 97.62% | 98.4% | 86.32% | 93.43 |
| Avg. Imp. | 0.62% | 7.11% |

N/A: Not Available; Avg. Imp.: Average Improvement
between parameters and payloads in web requests, both forward and backward.

C. COMPARISONS WITH PREVIOUS STUDIES
Comparisons of anomaly-based studies using CISC and ECML/PKDD datasets with the proposed models are presented in Table 11. Detection rates of the proposed models that are used in Table 11, consist of the highest values of 10-fold cross-validation and percentage split. It can be seen that the most successful models in both binary-class and multi-class classification are the proposed Bi-LSTM with DA-DANA based model.

When the studies are examined, we can observe that deep learning techniques work successfully on web attack detection, in general. The best classification accuracy was provided for the CISC-2010 dataset with a 99.08% detection rate using the proposed model. In the binary-class classification of ECML/PKDD dataset, the proposed model had the highest detection rate, with 99.47%, while the secondary detection rate was 98.8% [34]. Thus a significant improvement was achieved in the detection of web attacks with the proposed model structure.

TABLE 11. Comparison of results with previous studies.

| Author          | Model     | CSIC Binary Class | ECML/PKDD Binary Class | Multi Class |
|-----------------|-----------|-------------------|------------------------|-------------|
| Nguyen and Franke [7] | Naïve Bayes | 72.78% | 85.12% | N/A |
| Nguyen and Franke [7] | Bayes Network | 82.79% | 86.95% | N/A |
| Nguyen and Franke [7] | RH Network | 72.46% | 87.69% | N/A |
| Nguyen and Franke [7] | Hedge Boosting | 82.10% | 86.30% | N/A |
| Nguyen and Franke [7] | A-ExIDS | 90.98% | 92.56% | N/A |
| Nguyen et al. [34] | C4.5 | 94.49% | 96.37% | N/A |
| Nguyen et al. [34] | CART | 94.12% | 96.11% | N/A |
| Nguyen et al. [34] | Random Tree | 92.30% | 96.89% | N/A |
| Nguyen et al. [34] | Random Forest | 93.71% | 98.80% | N/A |
| Tekerek et al. [25] | Hybrid Model | 94.50% | 93.30% | N/A |
| Risati et al. [35] | Static Method | 90.65% | 92.93% | 90.00% |
| Sahoo et al. [36] | N-activity | N/A | N/A | N/A |
| Liang et al. [18] | C4.5 | 96.30% | N/A | N/A |
| Liang et al. [18] | Modsecurity | 55.20% | N/A | N/A |
| Liang et al. [18] | Naïve Bayes | 84.08% | N/A | N/A |
| Kooz et al. [37] | AdaBoost | 83.28% | N/A | N/A |
| Kooz et al. [37] | J48 | 95.97% | N/A | N/A |
| Mao et al. [38] | Autoencoder | 98.23% | N/A | N/A |
| Hao et al. [17] | Bi-LSTM | 98.35% | N/A | N/A |
| Liang et al. [18] | RNN | 85.15% | N/A | N/A |
| Liang et al. [18] | GRU | 97.88% | N/A | N/A |
| Liang et al. [18] | LSTM | 98.42% | N/A | N/A |
| Tekerek et al. [15] | CNN | 97.07% | N/A | N/A |
| Zhang et al. [16] | CNN | 96.49% | N/A | N/A |
| Ito and Iyama [59] | CNN | 98.80% | N/A | N/A |
| Vanzin et al. [40] | DNN | 97.24% | 94.33% | N/A |
| Nguyen et al. [41] | PL-RNN | 96.13% | N/A | N/A |
| Yu et al. [42] | RCNN | 97.91% | N/A | N/A |
| Yu et al. [42] | TextCNN | 98.48% | N/A | N/A |
| Current study | Bi-LSTM without DA-SANA | 98.35% | 98.94% | 87.66% |
| (Proposed Model) | Bi-LSTM with DA-SANA | 99.08% | 99.47% | 93.91% |

N/A: Not Available

There are few studies on multi-class classification for web attack detection. When the results of the existing studies and the proposed model are compared, it is clear that our classification success was impressively higher than the others, with a success rate of 93.91% for ECML/PKDD and 96.59% for our dataset (see Table 9). Since attack payloads are very similar, multi-class classification becomes very difficult in web attack detection. Moreover, because the datasets are imbalanced, the detection of the type of attacks becomes more laborious. With the DA-SANA method, more noise is added to some typical attack requests containing a long payload, so the sensitivity (TP rate) and the specificity (TN rate) are increased and this success detection rate is achieved. In addition, when results are compared with other models, the contributions and effectiveness of the novel DA-SANA Bi-LSTM model in the detection of web attacks are clearly seen.

D. TIME CONSUMPTION AND LIMITATIONS
The execution time required for the training of an HTTP request for three datasets is shown in Table 12. When these models have been trained once, they can be used directly for new HTTP requests and do not need to undergo the training process again for source efficiency, for a long time. Table 12 shows that the DNN model required the lowest time consumption for the two-class with 0.043 microseconds (ms), while the Bi-LSTM model required the highest time consumption with 1.8 ms. Similarly, the DNN model required the lowest time consumption for the multi-class with

---

**TABLE 12. Average execution time of training of a web request (in millisecond).**

| Model | Method | Dataset   | Average execution time (ms) |
|-------|--------|-----------|-----------------------------|
|       |        | Binary-Class | Multi-Class                 |
| Bi-LSTM |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 2.6 | 2.7 |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 2.3 | 2.4 |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 0.7 | N/A |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 1.8 | 2.5 |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 18.4 | 19.0 |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 15.2 | 18.7 |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 9.8 | N/A |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 14.4 | 18.8 |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 2.12 | 2.14 |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 2.11 | 2.07 |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 1.07 | N/A |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 1.76 | 2.1 |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 1.91 | 1.95 |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 2.13 | 1.9 |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 0.907 | N/A |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 1.64 | 1.92 |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 0.108 | 0.121 |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 0.116 | 0.118 |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 0.075 | N/A |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 0.099 | 0.119 |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 0.038 | 0.050 |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 0.046 | 0.055 |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 0.046 | N/A |
|       |      | OUR DATASET | ECML/PKDD | CSIC-2010 | 0.043 | 0.053 |
In the next study, the plan is to use hybrid deep learning methods in conjunction with DA-SANA to improve detection rates for attack types with a low F1-score. Additionally, since time cost is an important limitation of the proposed model, the next study will focus on solving this problem in addition to further improvements to the multi-class classification success. In future works, web anomaly detection can be performed using unsupervised methods based on autoencoder structure with DA-SANA, and unlabeled web server logs can be used, as well as HTTP requests.

REFERENCES

[1] Vulnerability Statistics Report, EdgeScan, Dublin, Ireland, 2019, pp. 4–17. [Online]. Available: https://www.edgescan.com/wp-content/uploads/2019/02/edgescan-Vulnerability-Stats-Report-2019.pdf

[2] Owasp. (2017). Top 10 Application Security Risks—2017. [Online]. Available: https://owasp.org/www-project-top-ten/2017/

[3] M. Sevri and H. Karacan, “Deep learning based web application security,” in Proc. 2nd Int. Conf. Adv. Technol., Comput. Eng. Sci. (ICATCES), Antalya, Turkey, Apr. 2019, pp. 349–354.

[4] P. Luptak, “Bypassing web application firewalls,” in Proc. 6th Int. Sci. Conf. Secur. Protection Inf., 2011, pp. 79–88.

[5] C. Kruegel and G. Vigna, “Anomaly detection of web-based applications,” in Proc. 10th ACM Conf. Comput. Commun. Secur. (CCS), Oct. 2003, pp. 251–261, doi: 10.1145/948109.948144.

[6] C. Liu, J. Yang, and J. Wu, “Web intrusion detection system combined with feature analysis and SVM optimization,” EURASIP J. Wireless Commun. Netw., vol. 2020, no. 1, pp. 1–9, Dec. 2020, doi: 10.1186/s13638-019-1591-1.

[7] H. T. Nguyen and K. Franke, “Adaptive intrusion detection system via online machine learning,” in Proc. 12th Int. Conf. Hybrid Intell. Syst. (HIS), Pune, India, Dec. 2012, pp. 271–277, doi: 10.1109/HIS.2012.6421346.

[8] N. M. Chang, “Improving efficiency of web application firewall to detect code injection attacks with random forest method and analysis HTTP request,” Program. Comput. Softw., vol. 46, no. 5, pp. 351–361, Oct. 2020, doi: 10.1134/S0361768820050072.

[9] Y. Zhou and P. Wang, “An ensemble learning approach for XSS attack detection with domain knowledge and threat intelligence,” Comput. Secur., vol. 82, pp. 261–269, May 2019, doi: 10.1016/j.cose.2018.12.016.

[10] D. Kar, S. Panigrahi, and S. Sundararajan, “SQLiGoT: Detecting SQL injection attacks using graph of tokens and SVM,” Comput. Secur., vol. 60, pp. 206–225, Jul. 2016, doi: 10.1016/j.cose.2016.04.005.

[11] B. A. Tama, L. Nkenyereye, S. M. R. Islam, and K.-S. Kwak, “An enhanced anomaly detection in web traffic using a stack of classifier ensemble,” IEEE Access, vol. 8, pp. 24120–24134, 2020, doi: 10.1109/ACCESS.2020.2969428.

[12] S. Park, M. Kim, and S. Lee, “Anomaly detection for HTTP using convolutional autoencoders,” IEEE Access, vol. 6, pp. 70884–70901, 2018, doi: 10.1109/ACCESS.2018.2881003.

[13] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, “A novel two-stage deep learning model for efficient network intrusion detection,” IEEE Access, vol. 7, pp. 30373–30385, 2019, doi: 10.1109/ACCESS.2019.2899721.

[14] M. A. Ferrag, L. Maglaras, S. Moschyiannis, and H. Janicke, “Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study,” J. Inf. Secur. Appl., vol. 50, Feb. 2020, Art. no. 102419, doi: 10.1016/j.jisa.2019.102419.

[15] A. Tekerek, “A novel architecture for web-based attack detection using convolutional neural network,” Comput. Secur., vol. 100, Jan. 2020, Art. no. 102096. Accessed: Apr. 11, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404820309706

[16] 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 Proc. Int. Conf. Neural Inf. Process., Cham, Switzerland: Springer, Nov. 2017, pp. 828–836, doi: 10.1007/978-3-319-70139-4_84.

[17] S. Hao, J. Long, and Y. Yang, “BL-IDS: Detecting web attacks using bi-LSTM model based on deep learning,” in Proc. Int. Conf. Secur. Privacy New Comput. Environ. Cham, Switzerland: Springer, Apr. 2019, pp. 551–563, doi: 10.1007/978-3-030-21375-2_45.
