Two Stage Deep Learning Based Stacked Ensemble Model for Web Application Security

Mehmet Sevri[1*], and Hacer Karacan[2]
[1] Informatics Institute, Gazi University
Ankara, Turkey
[e-mail: mehmetsevri@gmail.com]
[2] Computer Engineering Department, Faculty of Engineering, Gazi University,
06570 Ankara, Turkey
[e-mail: hkaracan@gazi.edu.tr]
[*]Corresponding author: Mehmet Sevri

Received March 30, 2021; revised January 3, 2022; accepted February 10, 2022; published February 28, 2022

Abstract

Detecting web attacks is a major challenge, and it is observed that the use of simple models leads to low sensitivity or high false positive problems. In this study, we aim to develop a robust two-stage deep learning based stacked ensemble web application firewall. Normal and abnormal classification is carried out in the first stage of the proposed WAF model. The classification process of the types of abnormal traffics is postponed to the second stage and carried out using an integrated stacked ensemble model. By this way, clients' requests can be served without time delay, and attack types can be detected with high sensitivity. In addition to the high accuracy of the proposed model, by using the statistical similarity and diversity analyses in the study, high generalization for the ensemble model is achieved. Within the study, a comprehensive, up-to-date, and robust multi-class web anomaly dataset named GAZI-HTTP is created in accordance with the real-world situations. The performance of the proposed WAF model is compared to state-of-the-art deep learning models and previous studies using the benchmark dataset. The proposed two-stage model achieved multi-class detection rates of 97.43% and 94.77% for GAZI-HTTP and ECML-PKDD, respectively.

Keywords: Anomaly detection, deep learning, ensemble learning, web application firewall, web security.

1. Introduction

Web applications have become the primary targets of attackers with their widespread use nowadays. Many automated vulnerability-scanning tools constantly scan new web applications to reveal and exploit vulnerabilities. Even attackers named script kiddies with little knowledge can perform very sophisticated and damaging web attacks using ready-to-use attacking tools. In addition, according to the analysis performed on all vulnerabilities, the rate of critical vulnerabilities in the web application layer is approximately four times in the network layer [1]. However, when the literature is examined, it is seen that there are much more network-based intrusion detection system (IDS) studies than web-based studies [2]. The Open Web Application Security Project (OWASP) publishes the Top-ten most critical web vulnerabilities periodically [3]. Most common web attacks in the current version of OWASP top-ten vulnerabilities [3] are 42% SQL injection (SQLi), 19% Cross-site scripting (XSS), 7% Remote code execution (RCE), and 5% sensitive file disclosure attacks [1]. Injections are the most critical vulnerabilities on the OWASP top list and cover attacks, such as SQLi and RCE. Sending unsanitized payloads to an interpreter as part of a command or database query causes injection attacks. Attackers execute unintended commands on the interpreter with malicious payloads using injection vulnerabilities for privilege escalation or to access data with improper authorization [4-6]. Similarly, XSS attacks are caused by insufficient filtering and sanitization of the JavaScript or HTML commands received from the clients. In addition to changing the appearance of web applications with XSS attacks, this may cause session hijacking of victims due to the stealing of cookies. XML external entity (XXE) vulnerability occurs when the XML parser is misconfigured. In particular, XXE attack can lead to the disclosure of hidden system files, server-side request forgery (SSRF) vulnerability, denial of service and other system problems. File inclusion vulnerabilities occur when internal or external files and paths are included to the web application without being properly sanitized. Local File Inclusion (LFI) vulnerability causes the disclosure of sensitive files in the system, while Remote File Inclusion (RFI) causes the inclusion of external harmful files or paths to the system. Server Side Includes (SSI) attack adds dynamic content to static web pages, allowing malicious commands or scripts to run on the system. Web vulnerabilities can generally cause critical problems such as the disclosure or the corruption of sensitive data, privilege escalation, seize of victims' information or authorizations, denial of service or hacking of web servers.

Preventing these threats is a very important challenge, and in general, the following three different approaches can be applied.

- Direct approach: Developing web applications in a completely secure way during all the development phase,
- Testing approach: Detecting vulnerabilities in the web application by penetration tests, black-box and white-box tests, fuzzing or static/dynamic analysis of codes,
- Protective approach: Detecting attacks with web application firewalls (WAF),

Neither of these approaches alone is enough to provide full web security. All three approaches should be used together to ensure the security of critical web applications. Among all, the use of WAF is the most effective method for detecting and preventing web attacks. Network-based IDS (NIDS) cannot protect against web vulnerabilities. However, WAF systems detect attacks by deeply analysing the content of web requests, unlike NIDS. Web anomaly detection is a very laborious problem because the characteristics of web requests are generally very similar to each other [7]. Due to the nature of attack detection, the data have highly imbalanced distributions. Therefore, the development of an anomaly-based WAF system, which can classify rare types of attacks with high precision, is an important challenge

[4, 6]. Binary-class attack detection can be performed with high accuracy with a single algorithm-based model. However, multi-class attack detection is particularly prone to the high variance (over-fitting); therefore, generalization problem occurs. Also, it is difficult to classify the uncommon attack types because of the high bias (under-fitting) caused by the imbalanced distribution, so it is obvious that it will be very difficult to detect multi-class web anomalies in real-time with a single algorithm based classification model. An ensemble model combining weak base models in multi-class web anomaly detection will be more compatible with the real world. The primary task of WAF systems is to detect whether incoming requests are attack or normal web requests. However, to analyse the attack traffic, the type of attack traffic should be classified and reported to the system administrator. This allows system administrators to more easily identify the target and purpose of the attack, and to patch any vulnerability that is being exploited. However, performing multi-class classification of attack type together with intrusion detection (normal-abnormal classification) negatively affects the success and time performance of anomaly-based WAF systems.

In this study, a deep learning based stacked ensemble WAF model is developed that can detect web attacks in two stages. In the first stage of the developed WAF model, web requests are classified as normal and abnormal in binary-class. In the second stage, multi-class classification is performed with the stacked ensemble model. In the second stage, the requests classified as abnormal in the first stage are processed and the types of attacks are detected. Thus, the complexity arising from multi-class detection is transferred to the second stage, so web applications continue to serve without a time delay. Deep learning-based models are used in both stages of the developed WAF system.

Some of the main contributions of this study are summarized below:

- Normal requests do not need to be processed by the sophisticated model required by multi-class classification with the two-stage structure in the study. Thus, normal requests are saved from time delay and high false positive (FP), classified in a fast and high accuracy.
- Attack types are classified with high precision and accuracy thanks to the integrated stacked ensemble model used in the study.
- A very comprehensive web anomaly data set with a high ability to generalize called GAZI-HTTP is created.
- The models developed in the study are compared with previous studies using the benchmark ECML-PKDD dataset. In addition, by using GAZI-HTTP and ECML-PKDD datasets, models based on a single algorithm are developed and compared with the proposed model using DNN, CNN, GRU, and LSTM, which are among the most widely used deep learning models in cyber-security. To the best of our knowledge, the proposed model has the best results in detecting web attacks.
- The GAZI-HTTP dataset and source codes created within the scope of the study will be publicly shared with other researchers to enable reproducibility and contribute to future research*.

  * https://github.com/mehmetsevri/Two-Stage-Deep-Learning-Based-Stacked-Ensemble-Model-for-Web-Application-Security

The remainder of this paper is organized into five sections as follows. In the next part of the study, related studies on web security are discussed briefly. In the third section, preliminaries about feature construction from web requests, deep learning algorithms and diversity analyses used in the study are presented. In the fourth section, detailed information about the methodology of the study and the general structure of the proposed model are

presented. The performance evaluations of the experimental studies carried out in the fifth section are presented with discussions. Besides, the performances of the proposed model are compared to similar studies and state-of-the-art deep learning models. The last section consists of the conclusions and future research.

2. Related Work

In this section, important studies and methods used in the field of web security, especially anomaly-based studies are discussed. Two main methods are used in the development of WAF systems, namely signature-based and anomaly-based. In the field of web security, intrusion detection is carried out with four different approaches derived from the two main methods given above. These are signature-based (misuse) [4, 5, 7, 8], anomaly-based [9-13], policy-based [14], and hybrid attack detection systems [15]. The advantages, disadvantages, and challenges of the WAF approaches are presented in **Table 1**.

Table 1. The advantages, disadvantages, and challenges of WAF development approaches.

Approaches	Advantages	Disadvantages	Challenges
Signature-based WAFs [4, 5, 7, 8]	Very effective in detecting known attacks	Failing in detecting newly emerging attacks; Bypassing using encoding techniques [13]	Updating the signature database
Anomaly-based WAFs [9-13]	Effective in detecting known and newly emerging attacks	High FP rate; high variance (overfitting) and high bias (underfitting) [16]	Feature extraction and selection; Generalization for real-world
Policy-based WAFs [14]	Elimination of the disadvantages of signature and anomaly-based systems	Vulnerabilities due to incorrect sequencing of policies [14]	Requirement for domain experts, Difficulties in the operation of policies at large-scale systems
Hybrid WAF systems [15]	Combining the power of signature-based and anomaly-based systems	The disadvantages of signature-based and anomaly-based systems	Requirement for domain experts to the management of security rules [15]

Two different web security studies have been carried out by Nguyen et al. [9, 10], one of which is based on feature selection, and the other is based on an ensemble model, using different traditional machine learning algorithms. The authors [10] developed WAF models based on generic feature selection (GeFS) and different machine learning. The authors measured the performance of different models by manually extracting 30 different features from web request, and reducing features with the proposed GeFS. In their studies, an average detection rate of 97.04% was achieved by removing 63% of the features with GeFS for binary-class four traditional machine learning algorithms on the ECML-PKDD dataset. In the second study [9], the authors proposed an adaptive learning binary-class ensemble (A-ExIDS) WAF model that combines different IDS outputs with weight values. The A-ExIDS method achieved higher success than major voting and Hedge / Boosting ensemble combining methods and a 92.56% detection rate was achieved for ECML / PKDD. Similarly, Tama et al. [17] proposed a stacked ensemble WAF model in which different ensemble models were used instead of using machine learning algorithms as base learners. The authors performed web anomaly detection with a stack of ensemble models combining the outputs of three different ensemble models. The authors used random forest, gradient boosting, and XGBoosting ensemble models as base classifiers and combined the outputs with the generalized linear model (GLM). When

the results were examined, it was seen that the performance of some base models was higher or very close to the proposed model and the proposed model did not provide a significant improvement [17]. In addition, while some of the models used as base learners were nonlinear, the generalization would be poor since the outputs were combined with a linear model. Luo et al. [18] proposed a method that sequentially combine the XGBoosting and the positive-unlabelled (PU) learning model based on payloads in web requests. The authors first send the payloads to the ensemble model by vectorising them according to ASCII values, and then normal requests are sent to the PU learning model to detect unknown attacks. Since only payloads are taken into account in the proposed model by the authors [18], attacks in different parts of the web requests such as directory traversal or XXE cannot be detected.

Vartonui et al. [13] developed binary-class web anomaly models based on stacked-autoencoder and DBN-based feature selection and one-class algorithms. The authors used N-gram for feature extraction from web requests. The authors achieved a detection rate of 84.13% in model tests performed on the ECML-PKDD dataset. In another similar web anomaly study [11], the authors showed that the classification performances are increased by using the word embedding method in single deep learning models based on CNN and DNN. However, the weighted average of detection rates for attack types in tests performed on ECML-PKDD is calculated as 84.0% [11]. This demonstrates that a single model cannot achieve sufficient success in classifying web attack types.

Catak developed a model for the malicious network traffic classification that relies on 2-layer classification models to reduce time consumption [19]. In these presented studies in which multiple attack type classification is carried out, web request attack detection is generally performed with a single-stage multi-class classification model. However, since the main purpose of WAF systems is to classify attack and normal traffic, it will be more effective to simplify the problem by applying the divide and conquer technique and performing binary-class detection in the first stage. The main motivation for the realization of this study is the creation of a two-stage structure in which the complexity of the attack type detection process is postponed to the second stage in order not to affect the real-time performance of anomaly detection of the deep learning-based WAF system. An important difference of this study from the previous ones is the creation of a two-stage model. In this way, it is ensured that web applications work at a speed that can be used in the real world. In addition, thanks to the stacked ensemble model which combines robust base learners developed for the classification of attack types, higher accuracy and sensitivity are achieved compared to other studies. Moreover, while creating the ensemble model, base learners providing high diversity are selected with statistical methods and high generalization power is aimed for real-world systems. Our study differs from other studies in that we use a real-world dataset, namely GAZI-HTTP.

3. Preliminaries of Feature Extraction, Algorithms and Diversity Analyses

In this section, preliminary information about feature extraction from web requests, algorithms used in the study, and diversity analyses used in measuring model generalization power are given.

3.1. Feature Extraction

Artificial neural network (ANN) models do not accept text-based data as direct input. For this reason, encoding into numerical word vectors by means of feature extraction are required to

feed text-based web requests into ANN-based network. Feature extraction from text-based data is carried out in two main steps: word tokenization and word embedding. In the first step, numeric word tokens must be extracted from character and character sets with the word tokenization process. In the second stage of feature extraction, the word embedding process, which provides the representations of the word tokens in similar web requests, should be performed.

Since web attack types involve certain patterns, feature construction becomes important. When the web security studies in the literature are examined, it is possible to determine the features manually or to use frequency-based methods such as N-gram [4, 13] and TF-IDF [5]. Although it is simple to extract features with N-grams, it has significant disadvantages [20]. N-gram cannot represent the meanings and relationships of the patterns in web requests. In addition, determining the value of n is an important challenge. Whereas the meaningful content is not sufficiently represented in small n values, high n values cause the sparsity problem. In the TF-IDF method, the frequency of the words and the coexistence of the words are calculated. However, the relationships between words and their semantic structures are not considered. With state-of-the-art Bag-of-words techniques, such as Word2Vec, fastText, Glove, and Keras, important semantic structures are covered. Therefore, very important achievements are provided in Natural Language Processing (NLP) problems. With the use of bag-of-words techniques, meaningful connections are created between words in web requests, especially URL and payload parts. Within the scope of this study, the Keras tokenizer is used in the creation of word tokens. Keras word embedding, which focuses on the frequency and co-occurrences of words in the web request, is used in the study. Thus, powerful features, used in the classification of web requests, are constructed.

After feature extraction, feature selection is one of the most important challenges affecting success in traditional machine learning algorithms. Deep learning algorithms handle feature selection and dimension reduction processes thanks to their structure. Deep learning models remove insignificant features in each layer by reducing their weights and ensure that the significant features are transferred to the next layer. Since deep learning-based models are used within the scope of this study, the feature selection or dimensionality reduction process is not carried out.

3.2. Ensemble Learning

In classical learning, the most successful classifier model is determined by creating separate models for a problem by using different state-of-the-art algorithms. There are problems where algorithms can work well, as well as challenges that they cannot solve. It is very difficult to classify datasets with big volume, multi-class and imbalanced distributions, via a single classifier. Ensemble learning is a method that enables decision-making by combining the outputs of more than one model. Ensemble learning is based on Surowiecki's [21] theory of collective intelligence. The four basic rules that a successful collective intelligence must contain are the diversity of opinion, independence, decentralisation, and aggregation [21]. An ensemble model can be realized by training a base model with the same structure with different sub-learning sets. At the same time, training models with different structures or algorithms with repetitive or non-repetitive sub-learning sets can realize an ensemble model. In this way, complex problems can be solved by ensuring cooperation in collective action [22]. To create a successful ensemble model, successful base models must be combined. However, it should be ensured that the outputs of the learners are lowly correlated with each other by using the base learners with high diversity. The performance and diversity of base learners are generally inversely proportional [22]. Base learners with high successes generally contain low diversity

and vice versa. Ensemble learning is generally based on combining the predictions of base classifier models and is shown with the following equations (1-3). $\boldsymbol{D_{train}}$ denotes training data, $\boldsymbol{h(X_i)}$ denotes classifier based on a specific algorithm, $\boldsymbol{X_i}$ denotes $\boldsymbol{i}$. inputs, and $\boldsymbol{Y_i}$ denotes their model output.

$$\textit{Training Data}: D_{train} = x_n, y_n; n = 1,2,...,N \tag{1}$$

$$\textit{Base Classifier}: L : D_{train} \rightarrow h(X), \textit{ where } h(X_i) : X_i \rightarrow \Upsilon_i \tag{2}$$

$$\textit{Ensemble Classifier}: \begin{Bmatrix} L_1 : D_{train} \rightarrow h_1(X_1) \\ L_2 : D_{train} \rightarrow h_2(X_2) \\ \dots \\ L_T : D_{train} \rightarrow h_t(X_T) \end{Bmatrix} \Rightarrow \{h_1(X_1) \cup h_2(X_2) \cup ...h_T(X_T)\} \tag{3}$$

There are different approaches for creating ensemble models; the most well-known are Bagging and Boosting. The bagging method is based on training base models based on the same algorithm with new sub-learning sets produced by random selection and substitution from the existing learning set. One of the most known and successful bagging algorithms is the Random forest. Bagging algorithms show high correlation and low diversity in high similarity datasets such as web requests, despite the random data selection [23]. Weak classifiers are used in the boosting method. Base learners work sequentially, and each base learner is trained to correct the error of the previous one. The most known boosting algorithms are Adaptive Boosting (AdaBoost), Gradient-Boosting, and XGBoost. Boosting algorithms are generally highly affected by noise [23]. Since attackers tend to encode attack payloads in general, boosting methods may be insufficient against these attacks.

In this study, a stacked ensemble model [24] based on different deep learning algorithms is used. The stacking technique can prevent bias towards base learners independent from the learning set [24]. All classification models can be used as a base learner in stacking models. Ensemble learning can be created with models trained with different parameters based on a single algorithm suitable for the problem, or it can be created with models based on different algorithms. The generalization feature is important for detecting emerging attack types in anomaly-based intrusion detection systems. The generalization features of ensemble models are directly related to the similarities and diversities of the base learners they contain. Generally, predictions of base learners based on the same algorithm have high similarity even if they are trained with different parameters and sub-datasets. The most effective method to ensure high diversity is using base learners based on different algorithms [25]. As such, it is desirable to use a suite of base learners that are structured in very different ways, ensuring that they make different predictions and, in turn, have less correlated prediction errors [25]. One of the main purposes of this study is to design a real-time WAF system with high diversity and high generalization ability. Therefore, to create a WAF system with a high generalization feature within the scope of our study, two suitable base learner models based on CNN and DNN with high accuracy were constructed.

3.3. Base and Meta Classifier Algorithms

Within the scope of the study, DNN and CNN algorithms are used in the first and second stages. In this section, a brief introduction to the algorithms used in the study is presented.

3.3.1 Deep Neural Network (DNN)

The DNN is the most basic DL algorithm and consists of interconnected artificial neurons in

many successive layers [26]. Although DNN has a non-complex architecture, it is a robust and state-of-the-art DL algorithm that works with high accuracy performance in many different areas. However, it is highly affected by model parameters and optimization methods. Training of DL models is generally more time-consuming than base ANNs. The computational complexity and time consumption of the DNN models are much lower than other DL architectures [26]. The DNN algorithm creates representations between the input vectors and the expected outputs by building links between neurons in layers based on weight values.

3.3.2. Convolutional Neural Network (CNN)

The CNN algorithm is one of the most widely used methods in the literature. Although it is generally used in the field of image processing, it can also be used effectively in NLP problems. CNN architecture can generate robust generalization and representation vectors by convolutions and pooling processes. The CNN architecture has a series of convolution layers that convolve inputs, provide feature maps and transfer them to the next layer. The pooling layers, which are frequently connected to the sequences of convolutional layers, realize the down-sampling and reduce the number of network parameters. Maximum or average pooling techniques are often used for the subsampling stage. Architecture is created in which the significant features are transferred to the next layer by applying convolution and pooling processes to the inputs. The CNN architecture tries to converge to the minimum error rate by updating the parameters with back-propagation [27]. Usually, at the end of the network, there is a fully connected layer followed by a classification layer.

3.4. Statistical Similarities and Diversity Analyses of Stacked Ensemble Models

The similarities, correlations, and the diversities of base models are the most effective factors for the generalization ability of the ensemble model. The ability of the ensemble model to simulate real-world problems depends on the high success and the diversity of the base models. However, the two factors are generally inversely proportional, when the successes of the base models increase, their diversity decreases, and vice versa. It is important to ensure the trade-off between performance and diversity when creating ensemble models.

There are many statistical similarity analysis methods used for binary models. However, to the best of our knowledge, there is no general method for the diversity analysis of multi-classifier and multi-class models. It requires the adaptation of existing binary similarity and diversity methods to multi-model and multi-class according to the problem. In this study, five different statistical methods are used to measure the similarity and diversity of the base models. These are Cohen kappa coefficients, Pearson correlation p-value, Yule Q statistics, Disagreement Measure (Diversity), and double-faults methods. The details of these methods used for the diversity of stacked models are given below.

Cohen Kappa Coefficients: The Cohen kappa coefficient $\boldsymbol{K_{i,j}}$ between the two classifiers is calculated as in (4) [28]. $\boldsymbol{\rho_0}$ denotes the relative observed agreements between the classifiers, and $\boldsymbol{\rho_e}$ denotes the hypothetical probability of chance agreements of the two classifiers. Within the scope of the study, kappa statistics of stacked models are measured as the average kappa scores of base classifiers. The average kappa coefficient $\boldsymbol{K}$ of an ensemble model consisting of $\boldsymbol{M}$ base classifiers is calculated as in (5) [28].

$$\kappa_{i,j} = \frac{p_0 - p_e}{1 - p_e} = 1 - \frac{1 - p_0}{1 - p_e} \tag{4}$$

$$\kappa = \frac{2}{M(M-1)} \sum_{i=1}^{M} \sum_{j=i+1}^{M} \kappa_{i,j} \tag{5}$$

Pearson correlation p-values: The Pearson coefficient used in the study is derived from measuring the similarity of binary classifiers in the literature, specific to multi-class anomaly detection. The p coefficient of the stacked ensemble model is calculated as the average of the binary *p* coefficients scaled by the number of labels and the number of base classifiers. Pearson's p values are calculated by using the following terms [29].

- N^{TT} : Total number of predictions classified correctly by both Classifier-1 and Classifier-2, together.
- N^{TF} : Number of predictions correctly classified by Classifier-1, while incorrectly classified by Classifier-2.
- N^{FT} : Number of predictions incorrectly classified by Classifier-1, while correctly classified by Classifier-2.
- N^{FF} : Total number of predictions misclassified by both Classifier-1 and Classifier-2, together.

The correlation coefficient $\rho_{i,j}$ between the two classifiers is calculated as in (6). The average correlation coefficient ρ of the stacked model based on the correlations of base model pairs is calculated as in (7) [29].

$$\rho_{i,j} = \frac{N^{TT}N^{TF} - N^{TF}N^{FT}}{\sqrt{\left(N^{TT}+N^{TF}\right)\left(N^{FT}+N^{FF}\right)\left(N^{TT}+N^{FT}\right)\left(N^{TF}+N^{FF}\right)}} \tag{6}$$

$$\rho = \frac{2}{M(M-1)}\sum_{i=1}^{M}\sum_{j=i+1}^{M}\rho_{i,j} \tag{7}$$

Yule's Q statistics: Q-statistics is an effective diversity analysis test based on the Chi-square distribution. The Q coefficient between the two classifiers is calculated as in (8) [28, 29]. The average Q coefficient for the ensemble model consisting of M base learners is calculated as in (9) [29].

$$Q_{i,j} = \frac{N^{TT}N^{TF} - N^{TF}N^{FT}}{N^{TT}N^{TF} + N^{TF}N^{FT}} \tag{8}$$

$$Q = \frac{2}{M(M-1)}\sum_{i=1}^{M}\sum_{j=i+1}^{M}Q_{i,j} \tag{9}$$

Disagreement Measure (Diversity): The Disagreement measure is calculated by the ratio of the opposite predictions by the two classifiers to the total number of predictions. The disagreement measure of the two classifiers is calculated as in (10). The average disagreement measure of the ensemble model consisting of M base classification models on the test set consisting of N elements is calculated as in (11) [28].

$$dis_{i,j} = \frac{N^{TF}+N^{FT}}{N^{TT}+N^{TF}+N^{FT}+N^{FF}} \tag{10}$$

$$dis = \frac{2}{M(M-1)}\sum_{i=1}^{M}\sum_{j=i+1}^{M}dis_{i,j} = \frac{2}{NM(M-1)}\sum_{i=1}^{M}\sum_{j=i+1}^{M}N^{TF}+N^{FT} \tag{11}$$

Double Faults: The fact that base classifiers misclassify with the same label is the factor that most negatively affects the success of the ensemble model. The calculation of double fault between two classifiers is shown in (12). The average of double faults for the ensemble model consisting of M base learners is calculated as in (13).

$$DF_{i,j} = \frac{N^{FF(same-class)}}{N^{TT}+N^{TF}+N^{FT}+N^{FF}} \tag{12}$$

$$DF = \frac{2}{M(M-1)}\sum_{i=1}^{M}\sum_{j=i+1}^{M}DF_{i,j} = \frac{2}{NM(M-1)}\sum_{i=1}^{M}\sum_{j=i+1}^{M}N^{FF(same-class)} \tag{13}$$

4. Methodology

Within the scope of this study, a new WAF that can detect web anomalies in two stages is developed. The architecture of the proposed model in the study is shown in **Fig. 1**. In the first stage of the proposed web anomaly detection method, HTTP requests are classified as normal or abnormal with the developed binary-class model. Binary-class classification is carried out with a fast and effective model based on DNN in the first stage. Normally classified requests are sent directly to the webserver for processing. Web requests detected as attacks are dropped from the network and a copy of the request is sent to the second stage. Normal clients are not affected by the time and performance costs arising from the complexity of classifying attack types. In the second stage, the attack types of requests, which are detected as anomaly before, are classified with a more sophisticated deep learning-based stacked ensemble model. The pseudocode for the detection of web anomalies with the proposed model is shown in **Table 2**.

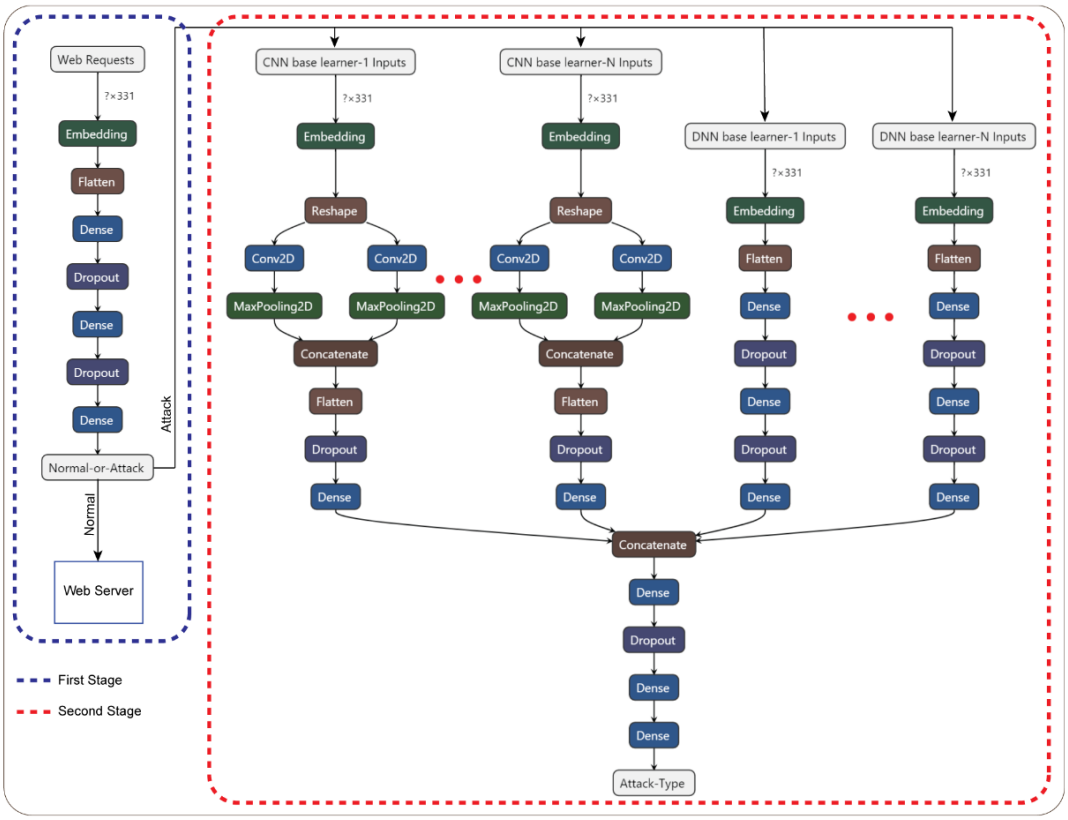

Fig. 1. The architecture of the two-stage DL based stacked ensemble model

Table 2. Pseudocode for detection of web anomalies with the proposed model

Input: Web requests $\Omega \leftarrow \{x_1, x_2, ..., x_m\}$ $\Omega \leftarrow \{(x_1, x_2 \ldots, x_m)\}$ with HTTP Request x,

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

Embedding vector $E_{m,n,k} \leftarrow$ processing word embedding using k embedding dim $\leftarrow \forall x_i \in \Omega$

Output: P_m; Normal tags or Attack types of given HTTP Requests

1. **for** $\forall R_i \in E_{m,n,k}$ $\forall x_i \in \Omega$ **do**
2. **Predict:** $P_i \leftarrow Model_{FIRST-STAGE}(R_i)$ detect normal or anomaly using the binary-class model
3. **if** $P_i = "Normal"$ **do**
4. send HTTP request x_i to the webserver for processing
5. **else do** // if HTTP request x_i is classified as "Abnormal" in the first-stage$\leftarrow N_i{*}C_i + S_i{*}C_i'$
6. **Drop:** drop x_i HTTP request from the network and send R_i to second-stage model
7. **Predict:** $P_i \leftarrow Model_{SECOND-STAGE}(R_i)$ detect the attack type using the stacked ensemble
8. **end for**

The flow chart of the proposed model in the study is shown in **Fig. 2**. As seen in **Fig. 2**, the development of the web anomaly detection model was carried out in two stages. Firstly, pre-processing of web requests and word tokenization operations were performed. In the first stage, a binary-class web anomaly detection model based on DNN was created by performing training and testing with web anomaly datasets. In the second stage, base learners were created by performing training and testing CNN and DNN models using only attack requests, and a meta learner model based on DNN was created by using predictions of these base learner models. The theoretical structures and mathematical representations of the models in the first and second stages are given below. Besides, the fundamental methods and techniques used in the development of models are explained.

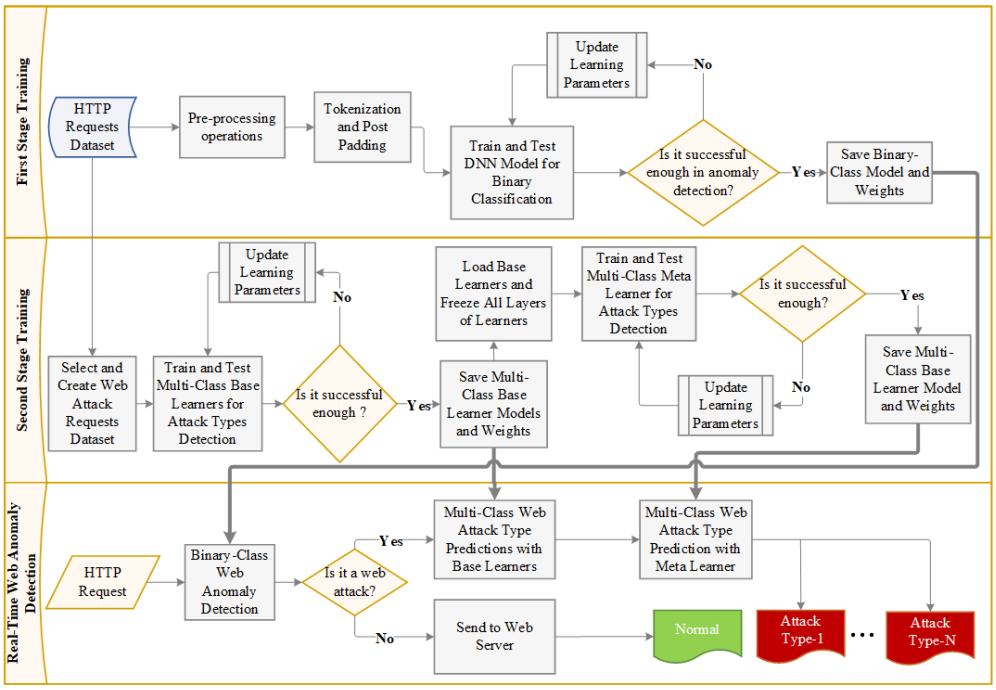

Fig. 2. The flow chart of the training and anomaly detection phases of the proposed model

4.1. First Stage: Binary-Class Web Attack Detection

Since web applications are time-sensitive systems, it is important to respond to normal clients' requests to the web application as quickly as possible. The first stage is very important in detecting and preventing attacks, and it is determined whether web requests are dropped or forwarded to the webserver. Therefore, it is aimed to develop a model with high classification success, low FP rate, and processing very fast. In the study, different binary-class models based on CNN, DNN, Gated Recurrent Unit (GRU), and Long Short Term Memory (LSTM) deep learning algorithms are trained and tested. The DNN model is determined as the fastest and most effective binary-class model for both datasets. It is seen that the highest computing power consumptions are required for LSTM and GRU based models, respectively. Within the scope of the study, firstly, cleaning of numeric and special characters except letters in web requests and data processing were carried out. At this stage, word sequences were created by separating words according to special characters and spaces in web requests. The words were converted into numerical values by tokenization process for entering the DNN model. The training process of the DNN model was carried out by using web requests with two different labels as normal and abnormal in datasets. In the first layer of the DNN model, there is the word embedding layer, which creates word vectors based on the relationships between the tokenized words. The architectural structure of the proposed DNN model for binary-class attack detection to be carried out in the first stage is shown in **Table 3**. To create the best model to be used in the first stage binary-class classification, a large number of models based on different numbers of layers and neurons have been tested. In our study, it has been determined that the model with DNN architecture in **Table 3** is the fastest model with sufficient performance for the first stage. The model consists of two DNN layers following the embedding layer and a prediction layer with Softmax at the output.

Fine-tuning is processed in the first stage for the optimization of the DNN model. To prevent over-fitting, dropout is applied with a 0.01 ratio after both DNN layers. Rectified Linear Unit (ReLU) is used as the activation function in the DNN layers. The Adam optimization algorithm with a learning rate of 0.001 is used for the optimization of the model. The binary cross-entropy loss function is used as the loss function. The batch size to be used in each iteration at the training stage is set to 1024.

Table 3. The architecture of the binary-class classifier model in the first stage.

Layer (Type)	Input Shape	Output Shape	Activation Function
Embedding	(m, n)	$(m, n, 8)$	-
Flatten	(m,n,8)	(m, n*8)	-
Dense-1	$(m, n * 8)$	$(m, 64)$	tanh
Dropout-1	$(m, 64)$	$(m, 64)$	-
Dense-2	$(m, 64)$	$(m, 64)$	tanh
Dropout-2	$(m, 64)$	$(m, 64)$	-
Dense	$(m, 64)$	$(m, 2)$	Softmax

4.2. Second Stage: DL Based Stacked Ensemble Model for Classification of Web Attack Types

In the second stage, attack types are classified, and models are trained using only web attack requests in datasets. The architectural details of the base models and the meta learner model used in the stacked ensemble model are given below.

4.2.1. Base Classifiers

In the creation of base learners, models in different variations are developed and tested with CNN, DNN, GRU, and LSTM deep learning algorithms. GRU and LSTM based learners negatively affect the performance and time consumption of the stacked ensemble model. Using CNN and DNN algorithms together is more efficient in detecting the types of attacks in terms of performance and time consumption. Therefore, base models are developed based on CNN and DNN architectures. The architectural structures of CNN and DNN base learners used in the study are shown in **Table 4**. Different stacked ensemble models are developed with varying numbers of base models. Stacked ensembles are created with 2, 4, 6, 8, and 10 base learners, respectively. The number of CNN and DNN learners is the same in a stacked ensemble model (e.g., Six base classifier model consists of three CNN and three DNN base learners). The performance and generalization ability of stacked ensemble models are tested with different statistical methods. Base learners are trained with non-repetitive different subsets of the learning set to avoid overfitting and high correlation.

CNN Base Learners: In this study, a CNN-based base learner is developed to classify the web attack type with a high detection rate. CNN base learners comprise concatenating the outputs of two structures consisting of the two-dimension convolution layer and the following max-pooling layer in a flatten layer. At the end of the base learners, there is a SoftMax layer for classification.

DNN Base Learners: In the study, a robust DNN base learner is created for the classification of web attack types. There is a flatten layer that reduces the three-dimensional vector outputs of the embedding layer to two dimensions in DNN base learners. In addition, DNN base learners consist of two hidden DNN layers and a SoftMax layer for classification, respectively.

Table 4. Architectures of base learner models in the second stage.

Base Learner	Layer (Type)	Input Layer	Input Shape	Output Shape	Act. Function
CNN	Embedding	Inputs	(m, n)	$(m, n, 16)$	-
	Reshape	Embedding	(m,n,16)	(m, n, 16, 1)	-
	Conv2D - 1	Reshape	$(m, n, 16,1)$	$(m, n, 1, 512)$	ReLU
	Conv2D - 1	Reshape	$(m, n, 1,512)$	$(m, n-1, 1,512)$	ReLU
	MaxPooling2D - 1	Conv2D - 1	$(m, n, 1,512)$	$(m, 1,1,512)$	-
	MaxPooling2D - 2	Conv2D - 2	$(m, n-1, 512)$	$(m, 1,1,512)$	-
	Concatenate	Pooling2D – 1 and Pooling2D - 2	$(m, 1,1, 512)$	$(m, 1,2, 512)$	-
	Flatten	Concatenate	$(m, 2,1, 512)$	$(m, 1024)$	-
	Dense	Flatten	$(m, 1024)$	$(m, 8)$	softmax
DNN	Embedding	Inputs	(m, n)	$(m, n, 16)$	-
	Flatten	Embedding	(m,n,16)	(m, n*16)	-
	Dense-1	Flatten	$(m, n * 16)$	$(m, 32)$	ReLU
	Dropout-1	Dense-1	$(m, 32)$	$(m, 32)$	-
	Dense-2	Dropout-1	$(m, 32)$	$(m, 32)$	ReLU
	Dropout-2	Dense-2	$(m, 32)$	$(m, 32)$	-
	Dense	Dropout-2	$(m, 32)$	$(m, 8)$	Softmax

4.2.2. Meta Learner

The proposed web attack type detection model is an integrated stacked ensemble model. In the integrated stacked ensemble model, a copy of the inputs is given to each base learner separately. Then the predictions of the base learners are concatenated and given as inputs to the meta-learner model. The stacked method enables the collection of models to conduct as a single large model. This approach is based on training the meta-learner on how to best combine base model predictions. Commonly used methods such as hard voting, soft (weighted) voting, linear regression, Naive Bayes (NB), and different deep learning algorithms are tested as meta learner. The DNN model, which combines the outputs of the base models most successfully, is used as a meta learner in the proposed stacked ensemble model. The meta-learner consists of two hidden DNN layers, and a SoftMax output layer, respectively.

The development of the proposed stacked ensemble model is carried out in two steps. Firstly, base learners are trained, and model parameters are recorded with weight values. Base learners are loaded and the stacked ensemble model is prepared for training by integrating the meta-learner. At this stage, all layers of the base learners are frozen and weight values are not allowed to change. Thereby, it is ensured that only the meta learner layers are trained. Model parameters and weight values are recorded after the meta-learner is trained with the training set reserved for it.

4.2.3. Model Optimization and Fine-Tuning for the Second Stage

Within the scope of this study, different optimization methods are used to increase the performance of base and meta learner models. The ReLU is used as the activation function in layers of base models. The Adam optimization algorithm with a learning rate of 0.001 is used in the optimization of the models. The categorical cross-entropy loss functions are used in the training of models. The batch size for each iteration is determined as 1024 in the training and testing of models. Finally, the fine-tuning of the ensemble is carried out to train all layers of ensemble models in the study. This process needs to be performed at a very low learning rate and few epochs. First of all, after the training of the meta-learner model is completed, all layers of the stacked model are de-frozen. Within the scope of the study, the fine-tuning process is carried out with 1×10^{-5} a learning rate in two epochs. Thus, it is ensured that base learners are well fitted with the meta learner.

5. Evaluations and Experimental Results

In this section, detailed information about the experimental results of the proposed method and performance evaluation with different measurement metrics are presented. First of all, the datasets used in the study are summarized, and experimental setups are explained. Afterward, performance and diversity analyses of the proposed method on two different web anomaly datasets are performed and the results are discussed. Finally, the proposed method is compared to previous studies and state-of-the-art DL models are developed in the current study.

5.1. Datasets

Network-based security studies and the number of public datasets in the literature are much higher than the number of studies and datasets for web application security. In some network-based datasets, it is seen that some types of web attacks are also covered by network attack types. However, this situation does not reflect the real-world situation, and it is not possible to examine web content with NIDS. There are very few public benchmark datasets in the field of

web application security. Existing datasets are also out-of-date and do not contain up-to-date attack patterns. When the literature is examined, it is seen that web security studies are mostly based on binary-class (normal and anomalous), and the number of studies based on multi-class is few. There are two public benchmark web anomaly datasets commonly used in the literature. One of them is CSIC-2010 [9] which is binary-class, and the other is multi-class ECML-PKDD [30] datasets. ECML-PKDD and CSIC-2010-HTTP were created in 2007 and 2010, respectively, and do not include current web attack payloads. It is important to use real-world data in the development and testing of intrusion detection systems. Within the scope of the study, a multi-class web anomaly dataset called GAZI-HTTP is created, which consists of up-to-date, comprehensive and real-world data.

In this study, ECML-PKDD and GAZI-HTTP datasets are used in the development, testing and the comparisons of the proposed models. Information about the datasets used in the study is given in detail below. The datasets are randomly divided into three sub-sets, 60% for the training of base learners, 20% for the training of the meta learner, and the remaining 20% for testing all developed models. Separating the train sets of the base learners from the meta learner is important for avoiding high variance. The binary-class model in the first stage is trained with an 80% subset used in training the base and meta learner models.

5.1.1. GAZI-HTTP Dataset

Within the scope of this study, a web anomaly dataset named GAZI-HTTP is created based on the real world, covering different attack types in OWASP Top-Ten [3]. Web traffics are collected by two methods via honeypot system and internal network in the study. In the first method, all web requests incoming to two different publicly configured web applications are recorded. One of these applications is a WordPress based web application that looks like a crypto coin trading platform. The second web application is designed as an X-Cart based e-commerce site. All requests to these web applications are tagged with the current OWASP Modsecurity CRS and recorded. In the second method, web requests in normal, and attack types are generated and sent to web applications hosted on internal web servers. In the second method, local copies of the same web applications are used. A large-scale and comprehensive web anomaly dataset is created by combining web requests from the real world via honeypots and internally generated web requests. The steps and details of the generation of the internal dataset are explained below.

Table 5. Distributions of GAZI-HTTP dataset.

Type	Base Learners Train-Subset	Meta Learners Train-Subset	Test Set	Total Size	Distribution
Valid	44117	14706	14706	73529	67.74%
SQLi	6482	2160	2161	10803	9.95%
XSS	5483	1828	1828	9139	8.42%
Command Injection	1941	647	647	3235	2.98%
LFI	1731	577	577	2885	2.66%
Open Redirect	1635	545	545	2725	2.51%
CRLF	1566	522	522	2610	2.40%
SSI	1178	393	392	1963	1.81%
XXE	994	331	331	1656	1.53%
Total	**65127**	**21709**	**21709**	**108545**	**100%**

Attack types and distributions in the GAZI-HTTP dataset are presented in **Table 5**. When the distribution is examined, it is seen that the GAZI-HTTP dataset is imbalanced. While 67.74% of web requests are "Valid", the remaining 32.36% consists of anomaly requests. In addition, the distribution of attack types is also imbalanced, in parallel with real-world attack types.

Internal web anomaly dataset: Firstly, two different web applications are installed on local servers. Web applications are scanned with a spider tool and normal web requests and parameters received are recorded. Web requests which include normal and attack payloads sent using original requests, and parameters are recorded in different databases and labelled according to the type of payload. Payloads in normal requests are selected from values such as integers, strings, or dates in suitable with the values received by the requests. Normal payloads are created with word clusters such as the most used words in the world, countries, cities, names, and surnames. The attack payloads used in the study are created by combining the attack payloads browsed on the internet, primarily GitHub, and the payloads used by open-source attack tools. The payload sets created in the study are very comprehensive and consist of attack types in the current OWASP Top-Ten list. To the best of our knowledge, there is no public web anomaly dataset as comprehensive and up-to-date as the GAZI-HTTP.

5.1.2. ECML-PKDD HTTP Dataset

This dataset was published at the ECML/PKDD conference in 2007, as a part of the Analysing Web Traffic ECML/PKDD 2007 Discovery Challenge [30]. There are 52296 HTTP requests in eight different classes, including “Valid” and seven different types of web attacks. The dataset is imbalanced and more than 66% of the data are labelled as “Valid”. The distribution of the labels in the dataset is shown in **Table 6**.

Table 6. Distributions of ECML-PKDD dataset.

Type	Base Learners Train-Subset	Meta Learners Train-Subset	Test Set	Total Size	Distribution
Valid	21003	7001	7002	35006	66.94%
Ldap Injection	1367	456	456	2279	4.36%
OS Commanding	2052	684	684	3420	6.54%
Path Traversal	1818	605	606	3029	5.79%
SSI	1139	380	379	1898	3.63%
SQLi	1536	512	512	2560	4.90%
XPath Injection	1367	456	456	2279	4.36%
XSS	1095	365	365	1825	3.49%
Total	**31377**	**10459**	**10460**	**52296**	**100%**

5.2. Experimental Tools

Kali Linux, Burp Suite, Paros, and TcpFlow are used for creating the GAZI-HTTP web anomaly dataset as web spiders, web attacking, and network sniffer tools. The Keras and the Tensorflow open-source platforms with Python programming language are used in the realization of the study. Additionally, panda, NumPy, scikit-learn, matplot libraries are used for data pre-processing, statistical analyses, 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 is used.

5.4. Results and Discussions

In this section, the experimental results of the proposed two-stage stacked ensemble model for web anomaly detection are presented and discussed in detail. The results of the models at each stage are discussed in separate sections. Firstly, the results of the binary-class DNN model in the first stage are presented. Secondly, the results of the base learners and the stacked ensemble model used for the attack type classification in the second stage are presented. In the last part, the results of the final combined two-stage model that classifies all web request types are presented and discussed.

5.4.1. First Stage: Binary-Class Classifier Results

Confusion matrices and performance evaluations of DNN models in the first stage based on the GAZI-HTTP and ECML PKDD test sets are shown in **Table 7**. The proposed DNN model achieves detection rates of 98.76%, 99.81% on the GAZI-HTTP and ECML-PKDD, respectively. FP rates are 0.72% and 0.11% for GAZI-HTTP and ECML-PKDD, respectively. When the performance of the proposed binary-class model is examined, it is seen that the model is not affected by imbalanced distribution and has very low FP rates. In addition, the high rates of bias-variance trade-off are provided with DNN model.

Table 7. Confusion matrices and performance evaluations of binary-class classification in the first stage.

			Predicted Class			**Performance Evaluations**			
	Dataset	Class	Normal	Abnormal	Total	Precision	Recall	F-Score	**ACC**
Actual Class	GAZI HTTP	Normal	**14600**	106	14706	0.9847	0.9766	0.9806	
		Abnormal	164	**6839**	7003	0.9889	0.9928	0.9908	**0.9876**
		Total / W. Avg.	14764	6945	21709	**0.9876**	**0.9876**	**0.9875**	
	ECML PKDD	Normal	**3454**	4	3458	0.9957	0.9988	0.9973	
		Abnormal	15	**6987**	7002	0.9994	0.9979	0.9986	**0.9981**
		Total / W. Avg.	3469	6991	10460	**0.9982**	**0.9982**	**0.9982**	

5.4.2. Second Stage: Stacked Ensemble Models Results

Five different stacked ensemble models are developed, consisting of 2, 4, 6, 8, and 10 base learners, half of which are CNN, and the remaining half are DNN models. Stacked ensemble models are compared with different statistical methods by evaluating their similarity, diversity, and performance. We aim to select the best stacked model that has high detection rates together with high generalization for the real-world WAF system. Classification performances and diversity analyses of stacked ensemble models are shown in **Fig. 3** and **Table 8**, respectively. The highest and lowest detection rates for the GAZI-HTTP are achieved by ensemble models which consist of two and ten base learners with 96.10% and 94.65%, respectively. It is seen that diversity rates increase with the number of base learners on the GAZI-HTTP dataset, in contrast to the detection rates.

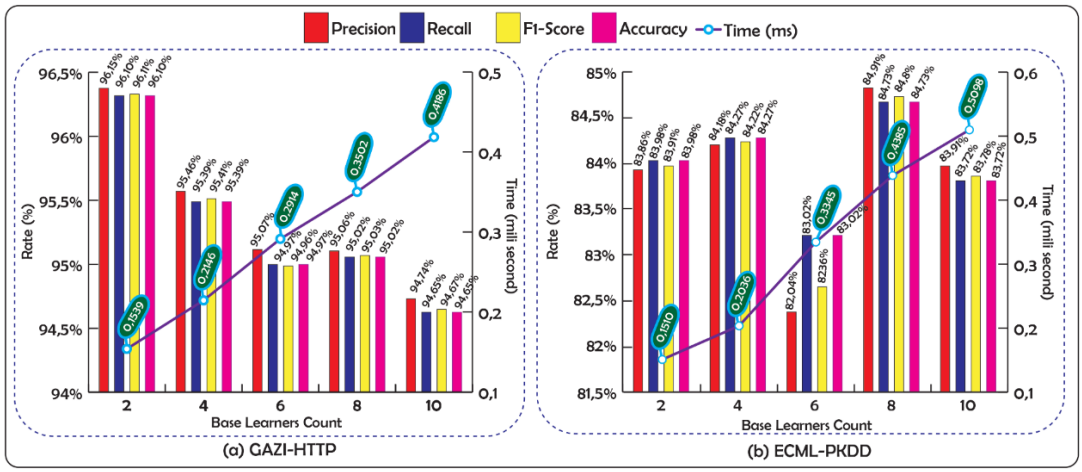

Fig. 3. Performance evaluations of stacked ensemble models

Table 8. Statistical diversity analyses of stacked ensemble models

Base Classifiers Counts		*Avg. Kappa Scores	*Avg. Pearson p-values	*Avg. Yule Q Similarity	Diversity	Avg. Double Fault
GAZI HTTP	2	0.9177	0.5579	0.9669	0.0431	**0.0298**
	4	0.8786	0.476	0.9222	0.0747	0.0387
	6	0.8424	0.4289	0.8831	0.0999	0.0448
	8	0.8124	**0.4149**	0.8561	0.1196	0.0522
	10	**0.7764**	0.4153	**0.8347**	**0.1413**	0.0665
ECML PKDD	2	0.5844	0.2364	0.5850	0.3212	**0.0925**
	4	0.4957	0.2239	0.5071	0.3655	0.1259
	6	0.4582	0.2317	0.5038	0.3729	0.1506
	8	0.4262	0.2129	0.4642	0.3926	0.1608
	10	**0.3937**	**0.2067**	**0.4441**	**0.4037**	0.1783

* It is inversely proportional to the diversity

The highest and lowest detection rates for the ECML-PKDD are achieved by ensemble models which consist of six and eight base learners with 83.02% and 84.73%, respectively. Similarly, it is seen that diversity increases with the number of base learners for the ECML-PKDD dataset. As the number of base learners' increases, the sub-training set gets smaller; therefore, weak base-classifier models become dissimilar. The avg. Kappa scores, avg. Pearson p-values and the mean Yule Q similarities metrics are inversely proportional to diversity, and while these metric values decrease, the similarity between base learners decreases and diversity increases. As seen in **Table 8**, the avg. Kappa scores, avg. Pearson p-values and mean Yule Q similarities decrease in direct proportion to the number of base learners. When **Table 8** is examined, it is seen that the statistical similarities and diversities of stacked ensemble models generally have the best divergence when the number of base learners is 10, and it consists of the most similar models when the number of base learners is 2.

The graph of detection rate and diversity trade-off for both datasets is presented in **Fig. 4**. When this graph is examined, it is seen that the classification performances and the diversities of the ensemble models in both datasets are generally inversely proportional. It is not the right approach to focus only on classification accuracies in ensemble models. To detect newly emerged web attacks, ensemble models should be developed with the ability to generalize; thus, have high diversity. When **Fig. 4** is examined, it is seen that the best accuracy-diversity

trade-offs for both datasets are provided by ensemble models consisting of eight base learners (shown in blue areas in **Fig. 4**). For this reason, we decide to use a stacked ensemble consisting of eight base learners with both a high detection rate and high diversity.

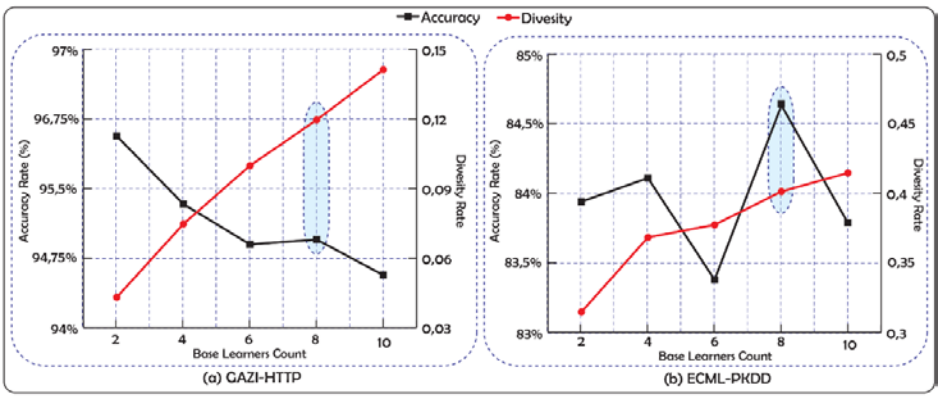

Fig. 4. Accuracy and diversity trade-offs of stacked ensemble models based on (a) GAZI-HTTP and (b) ECML-PKDD (blue areas show the best trade-off for the two datasets, together).

5.4.3. Two-Stage Model Results

The performances of the proposed two-stage model, which is formed by concatenating first stage and second stage models, are evaluated on test sets. The results below belong to the two-stage model, which consists of eight base learners. Performance evaluations of two-stage models based on GAZI-HTTP and ECML-PKDD datasets are presented in **Fig 5**. Detection rates of 97.43% and 94.97% are achieved for GAZI-HTTP and ECML-PKDD, respectively. The weighted average of precision, recall, and F1-score on GAZI-HTTP are achieved 97.44%, 97.43, and 97.43, respectively. Similarly, the average precision, recall, and F1-score on the ECML-PKDD test set are 94.84%, 94.77%, and 94.80%, respectively. It is seen that the proposed two-stage model generally has acceptable sensitivity and accuracy values for both datasets. When **Fig. 5** is analysed by class-based, the best F1-scores are reached for the "Valid" label in both datasets, and 99.08% and 99.86% for GAZI-HTTP and ECML-PKDD, respectively. The worst class-based classification performance for the GAZI-HTTP dataset is for the "Remote Code Execution" (RCE) label, with the F1-score remaining at 85.71%. Similarly, the class with the worst F1-score for ECML-PKDD is the "Server Side Include" SSI label that only reached 48.08%. The main reason for this is seen when **Tables 5** and **6**, which show the distribution of datasets, are examined. Both datasets are imbalanced, and it is seen that the classes that have few number requests reached the least performance. In addition, some web attack types may contain very similar payload structures. For example, 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.

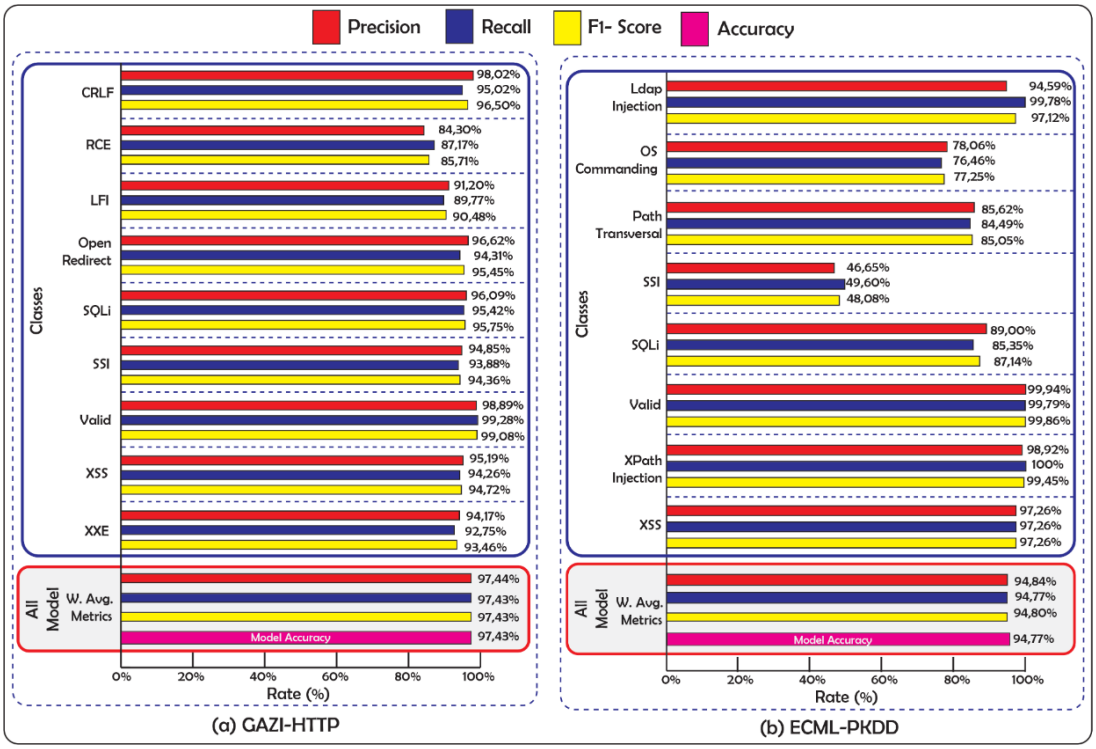

Fig. 5. Performance evaluations of two-stage models based on (a) GAZI-HTTP and (b) ECML-PKDD

5.4.4. Comparison with State-of-the-Art DL Algorithms and Previous Studies

Single classifier models based on DL algorithms, commonly used in web security, are developed for the purpose of performance comparisons with the proposed model in the study. DNN, CNN, LSTM, and GRU algorithms are used in the development of single classifiers. Multi-class performance comparisons of the proposed two-stage stacked ensemble model with DL models are shown in **Fig. 6**. Detection rates of 96.16%, 96.14%, 94.62% and 89.28% on GAZI-HTTP are achieved with the DNN, CNN, GRU and LSTM models, respectively. It is seen that the proposed model reached the highest detection rate, and a significant improvement is achieved with 97.43%. Similarly, detection rates of 88.76%, 90.20%, 86.84% and 86.57% are achieved for ECML-PKDD with the DNN, CNN, GRU and LSTM models, respectively. With the proposed model, 94.77% detection rate is achieved on ECML-PKDD, and 4.57% better classification success is achieved than the most successful single DL algorithm. It is obviously seen that the proposed model has significantly improved multi-class web anomaly detection compared to state-of-the-art DL models based on one algorithm.

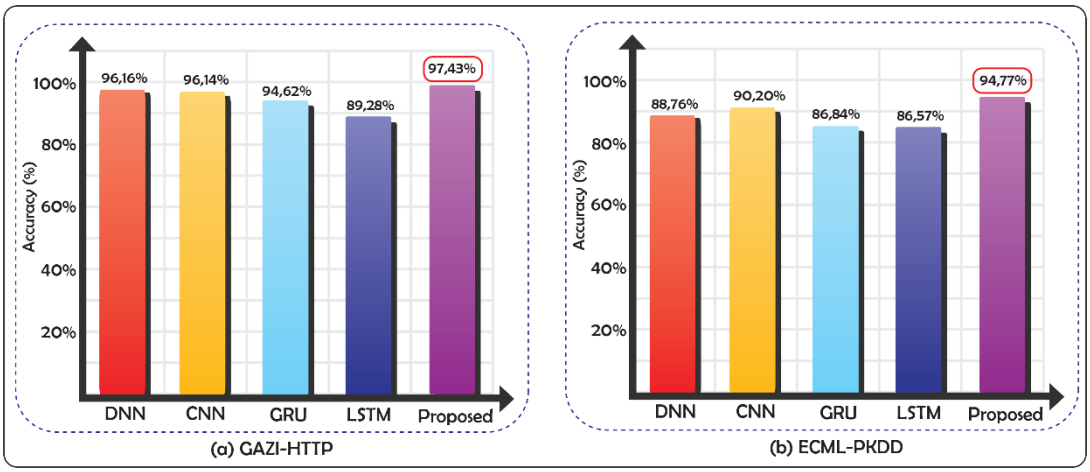

Fig. 6. Performance comparison of the proposed model with DL based one classifier models

Previous studies using the benchmark ECML-PKDD dataset are compared to the proposed model. When the literature is examined, it is seen that most of the studies are carried out based on binary-class, so multi-class studies are few. The comparison of the proposed model with other studies is shown in **Table 9**. It is seen that Nguyen and et al. [10] have reached the highest classification success so far with a detection rate of 98.80% for binary classification on ECML-PKDD. Since feature extraction is performed only by statistical methods in the study [10], the content of web requests is not considered semantically and is not suitable for a real-world system. When the results of the A-ExIDS model, proposed in study [9], are examined, it is seen that while high accuracy is provided for normal web requests, it is lower in detecting attack types. As a result, it causes high FP and is not suitable for WAF systems. It is seen that Vartouni et. al. [13] reached an 84.13% detection rate for ECML-PKDD. One of the important reasons for the low accuracy rate in the study [13] is the use of N-gram based feature extraction, which does not include semantic representations. When the results in the study conducted by Odumuyiwa et al [11] are examined in detail, the weighted average detection rate of the single classifier CNN model for attack types remains at 84.0%. Thus, it is seen that even if deep learning is used, attack types cannot be detected with high sensitivity with a single algorithm-based classifier, and high bias occurs due to an imbalanced dataset.

In our study, this rate is improved to reach 99.81% for binary-class. The single classifier based on CNN in the current study achieves the highest success among other models with a detection rate of 90.20% for multi-class on ECML-PKDD. The proposed two-stage stacked ensemble model achieves a detection rate of 94.77% by providing a 4.57% improvement on ECML-PKDD. It is seen that the proposed model is significantly more successful than other studies in detecting both binary-class and multi-class web anomalies.

Table 9. Comparison of results with previous studies on ECML-PKDD dataset

Author	*Model*	*Binary Class*	*Multi Class*	*Author*	*Model*	*Binary Class*	*Multi Class*
Nguyen and Franke [9]	Naive Bayes	85.12%	N/A	Raissi et al. [8]	Static Method	87.93%	69.00%
Nguyen and Franke [9]	Bayes Network	86.95%	N/A	Exbrayat [4]	N-similarity	90.00%	N/A
Nguyen and Franke [9]	RBF Network	87.69%	N/A	Pachopoulos et al. [12]	Decision Tree	77.00%	N/A

Nguyen and Franke [9]	Hedge Boosting	86.30%	N/A	Galagher et al. [5]	Tf-Idf	94.00%	N/A
Nguyen and Franke [9]	A-ExIDS	92.56%	N/A	Odumuyiwa et al. [11]	DNN	91.90%	N/A
Nguyen et al. [10]	Decision Tree	96.37%	N/A	Odumuyiwa et al. [11]	CNN	94.70%	N/A
Nguyen et al. [10]	CART	96.11%	N/A	Current Study	DNN	99.81%	88.76%
Nguyen et al. [10]	Random Tree	96.89%	N/A	Current Study	CNN	98.41%	**90.20%**
Nguyen et al. [10]	Random Forest	***98.80%***	N/A	Current Study	GRU	98.04%	86.84%
Tekerek et al. [15]	Hybrid Model	93.30%	N/A	Current Study	LSTM	97.84%	86.57%
Vartouni et al. [13]	DBN	84.13%	N/A	**Proposed Model**	**Two-Stage**	**99.81%**	**94.77%**

N/A: Not Available

5.4.5. The Time and Space Complexities of the Proposed Model

In the development of a real-time WAF system, time and space complexity is important in terms of computational resources and time latency. Deep learning algorithms require a significant amount of computation based on model depth and a number of parameters, in which intensive matrix multiplications are performed. In calculating the space complexity of deep learning algorithms, it should be taken into account since the outputs of neurons in each layer are transferred to the next layer and must be kept in memory. In addition, values of the parameters and the weights in the network should be stored in the memory. Therefore, the space complexity of the proposed model based on deep learning depends on the number of layers and parameters it contains and is linear. However, in proportion to the network size, the memory space allocation that the deep learning model will keep in memory depends on the number of inputs, the number of outputs of neurons, and the number of parameters included in the model, and space complexity is still linear [31].

The time complexity changes asynchronously depending on the number of inputs, the number of layers, the number of neurons in the layers, the number of outputs of the neurons, the initial values of the neuron weights and parameters, the loss function, the number of training epochs or the stopping criterion [31]. To absolute calculate the time complexities of deep learning-based models, it is necessary to determine how many neuron connections are active during the training of the DL network created. However, the number of active neuron connections can constantly change according to the changing learning parameters in a deep learning network. Since the proposed WAF model has two stages, it is processed only in the first stage or in both stages, depending on whether the request is normal or attack. Normal requests are processed only in the binary-class DNN model in the first stage. Bienstock et. al. [32] proposed a method to calculate the general time complexity of different deep learning architectures with their experimental study. The time complexity of the binary-class DNN model in the first stage of the method we propose based on this method [32] is seen in (14), where $\boldsymbol{m}$ denotes the input dimension, $\boldsymbol{n}$ denotes the output dimension, $\boldsymbol{k}$ denotes the number of layers, $\boldsymbol{N}$ denotes the total number of parameters, $\boldsymbol{D_{train}}$ and $\boldsymbol{D_{test}}$ are element number of the train and test sets and $\boldsymbol{E_{epoch}}$ is the number of training epochs.

$$O\left(\left(m\log(m)w^{O\left(k^2\right)}\right)^{n+m+N}\left(D_{train}+D_{test}\right)E_{epoch}\right) \tag{14}$$

Similarly, the general time complexity of the proposed second-stage attack type detection model, which includes base and meta learner models based on CNN and DNN models, is also seen in (15), where $\boldsymbol{M_{dnn}}$ is the number of meta learners (one meta learner was used in the study), $\boldsymbol{B_{dnn}}$ and $\boldsymbol{B_{cnn}}$ are the numbers of base learners based on DNN and CNN algorithm, respectively [32].

$$O\left(\left(m\left(\left(B_{dnn}+M_{dnn}\right)*\log\left(m\right)+B_{cnn}\right)w^{O\left(k^2\right)}\right)^{n+m+N}\left(D_{train}+D_{test}\right)E_{epoch}\right) \tag{15}$$

In addition to the time complexity above, the average time requirements of the web requests in the test set were calculated in order to see the time efficiency of the proposed model. Also, the time performances of the models based on a single deep learning algorithm and the proposed model were compared. In addition to the improvement of detection rate with the proposed model, reducing latency for normal web requests is one of the important aims in this study. Weighted average detection times of the proposed model for the first stage, second stage, two-stage and comparisons with models based on single DL algorithms are presented in **Table 10**. In the first stage, it takes an average of 0.041 and 0.044 milliseconds (ms) to classify a request for GAZI-HTTP and ECML-PKDD, respectively. In the second stage, the classification of attack types takes 0.339 and 0.847 ms, respectively. The average time consumption of multi-class detection of an unknown request in two stages takes 0.129 and 0.137 ms for GAZI-HTTP and ECML-PKDD, respectively. It is obviously clear that the delay of normal web requests due to anomaly detection is prevented with the proposed two-stage model. The computer used in the study can process approximately 25000 normal user requests per second. Moreover, the number of web requests that can be processed per unit time can be increased with a more powerful machine to be positioned in inline mode.

Table 10. Time performances of the proposed model and comparisons with WAF models based on a single algorithm

		Single classifier DL models				**Proposed model**		
	Dataset	DNN	CNN	GRU	LSTM	First Stage	Second Stage	W. avg. of two-stage
Time (ms)	GAZI HTTP	0.035	0.094	12.62	14.97	0.041	0.339	0.129
	ECML PKDD	0.048	0.138	12.849	15.634	0.044	0.847	0.137

6. Conclusions

In this study, a two-stage deep learning-based stacked ensemble model is proposed for effective web anomaly detection. In the first stage of the proposed model, normal web requests and attack requests are classified with a robust binary-class DNN model. In the second stage, the types of attack requests are classified with the DL based stacked ensemble model. Thanks to the proposed two-stage structure, the multi-class classification complexity used to determine the attack type is transferred to the second stage. Thus, normal users' requests can be processed with high accuracy and speed. Statistical diversity analyses are carried out to determine the generalization power of the proposed model in the study. As a result of the analyses, the stacked ensemble model with both high accuracy and high diversity is determined. With this study, we propose a model that classifies web requests based on multi-class with high sensitivity and without time delay for normal clients. With the proposed model, improvement is provided for GAZI-HTTP and ECML-PKDD by 1.27% and 4.57%, respectively, compared to other models, and the detection rates of 97.43% and 94.77% are achieved, respectively. It is seen that the proposed model has a significant performance improvement compared to the

state-of-the-art DL algorithms and previous studies. In addition, a comprehensive and robust web anomaly dataset named GAZI-HTTP is created within the scope of the study. In future studies, it is planned to develop hybrid DL based models in which attack types are handled separately. Also, the use of base learners based on manual feature extraction together with the base learners based on word embedding in the ensemble model will increase the power of generalization. Applying techniques such as resampling or data augmentation to address the problem of low classification performance for some classes resulting from imbalanced datasets will increase sensitivity and accuracy.

References

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

[2] M. Sevri, and H. Karacan, "Deep learning based web application security," in *Proc. of 2nd Int. Conf. on Advanced Technologies, in Proc. Computer Engineering and Science (ICATCES)*, Antalya, Turkey, pp. 349-354, Apr. 2019. Article (CrossRef Link)

[3] Owasp, "Top 10 Application Security Risks," 2017. [Online]. Available: https://owasp.org/www-project-top-ten/2017/

[4] M. Exbrayat, "ECML/PKDD challenge: analyzing web traffic a boundaries signature approach," in *Proc. of 18nd Int. Conf. ECML/PKDD*, Warsaw, Poland, pp. 53-64, Sep., 2007. Article (CrossRef Link)

[5] B. Gallagher, and T. Eliassi-Rad, "Classification of HTTP attacks: a study on the ECML/PKDD 2007 discovery challenge," 2009. Article (CrossRef Link)

[6] L. Liu, P. Wang, J. Lin, and L. Liu, "Intrusion Detection of Imbalanced Network Traffic Based on Machine Learning and Deep Learning," *IEEE Access*, vol. 9, pp. 7550-7563, 2020. Article (CrossRef Link)

[7] C. Kruegel, and G. Vigna, "Anomaly detection of web-based attacks" in *Proc. of 10th ACM Conf. on Computer and Communications Security*, pp. 251-261, Oct. 2003. Article (CrossRef Link)

[8] C. Raissi, J. Brissaud, G. Dray, P. Poncelet, M. Roche, and M. Teisseire, "Web analyzing traffic challenge: description and results," in *Proc. of 18nd Int. Conf. ECML/PKDD*, Warsaw, Poland, pp. 47-52, Sep., 2007. Article (CrossRef Link)

[9] H.T. Nguyen, and K. Franke, "Adaptive Intrusion Detection System via online machine learning," in *Proc. of 12th Int. Conf. on Hybrid Intelligent Systems (HIS)*, pp. 271-277, Dec. 2012. Article (CrossRef Link)

[10] H.T. Nguyen, C. Torrano-Gimenez, G. Alvarez, K. Franke, and S. Petrovic, "Enhancing the effectiveness of web application firewalls by generic feature selection," *Logic Journal of IGPL*, vol. 21, no. 4, pp. 560-570, Aug. 2013. Article (CrossRef Link)

[11] V. Odumuyiwa, and A. Chibueze, "Automatic Detection of HTTP Injection Attacks using Convolutional Neural Network and Deep Neural Network," *J. of Cyber Security and Mobility*, vol. 9, No. 4, pp. 489-514, 2020. Article (CrossRef Link)

[12] K. Pachopoulos, D. Valsamou, D. Mavroeidis, and M. Vazirgiannis, "Feature extraction from web traffic data for the application of data mining algorithms in attack identification," in *Proc. of 18nd Int. Conf. ECML/PKDD*, Warsaw, Poland, pp. 65-70, Sep., 2007. Article (CrossRef Link)

[13] A.M. Vartouni, M. Teshnehlab, and S.S. Kashi, "Leveraging deep neural networks for anomaly-based web application firewall," *IET Information Security*, vol. 13, no. 4, pp. 352-361, 2019. Article (CrossRef Link)

[14] L. Kagal, T. Finin, and A. Joshi, "A policy based approach to security for the semantic web," in *Proc. of 2nd Int. Semantic Web Conf.*, Sanibel Island, FL, USA, pp. 402-418, Oct., 2003. Article (CrossRef Link)

[15] A. Tekerek, C. Gemci, and O.F. Bay, “Design and implementation of a web-based intrusion prevention system: a new hybrid model,” *J. of the Faculty of Engineering and Architecture of Gazi University*, vol. 31, no. 3, pp. 646-655. 2016. Article (CrossRef Link)
[16] C. Hwang, D. Kim and T. Lee, "Semi-supervised based Unknown Attack Detection in EDR Environment," *KSII Transactions on Internet and Information Systems*, vol. 14, no. 12, pp. 4909-4926, 2020. Article (CrossRef Link)
[17] B. A. Tama, L. Nkenyereye, S. 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. Article (CrossRef Link)
[18] Y. Luo, S. Cheng, C. Liu and F. Jiang, "PU Learning in Payload-based Web Anomaly Detection," in *Proc. of 3rd Int. Conf. on Security of Smart Cities, Industrial Control System and Communications (SSIC)*, Shanghai, China, pp. 1-5, Oct., 2018. Article (CrossRef Link)
[19] Catak FO, “Two-layer malicious network flow detection system with sparse linear model based feature selection,” *Journal of the National Science Foundation of Sri Lanka*, vol. 46, no. 4, pp. 601-612, 2018. Article (CrossRef Link)
[20] A. Al-Alyan and S. Al-Ahmadi, "Robust URL Phishing Detection Based on Deep Learning," *KSII Transactions on Internet and Information Systems*, vol. 14, no. 7, pp. 2752-2768, 2020. Article (CrossRef Link)
[21] J. Surowiecki, “The wisdom of crowds,” *Anchor*, Aug. 2005. Article (CrossRef Link)
[22] K. Guzel and G. Bilgin, "Classification of Nuclei in Colon Cancer Images using Ensemble of Deep Learned Features," in *Proc. of Medical Technologies Congress (TIPTEKNO)*, Izmir, Turkey, pp. 1-4, Oct., 2019. Article (CrossRef Link)
[23] T.G. Dietterich, “An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization,” *Machine Learning*, vol. 40, pp. 139–157, Aug. 2000. Article (CrossRef Link)
[24] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241-259, 1992. Article (CrossRef Link)
[25] C. Zhang and Y. Ma, (ed.), *Ensemble machine learning: methods and applications*, Springer Science & Business Media, 2012. Article (CrossRef Link)
[26] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85-117, 2015. Article (CrossRef Link)
[27] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436-444, 2015. Article (CrossRef Link)
[28] Y. Bi, "The impact of diversity on the accuracy of evidential classifier ensembles," *Int. J. of Approximate Reasoning*, vol. 53, no. 4, pp. 584-607, 2012. Article (CrossRef Link)
[29] S.S. Choi, S.H. Cha, and C.C. Tappert, "A survey of binary similarity and distance measures," *J. of Systemics, Cybernetics and Informatics*, vol. 8, no. 1, pp. 43-48, 2010. Article (CrossRef Link)
[30] ECML/PKDD, "Analyzing Web Traffic ECML/PKDD 2007 Discovery Challenge," in *Proc. of 18nd Int. Conf. ECML/PKDD*, Warsaw, Poland, Sep., 2007. Article (CrossRef Link)
[31] G. Serpen and G. Zhenning, “Complexity analysis of multilayer perceptron neural network embedded into a wireless sensor network,” *Procedia Computer Science*, vol. 36, pp. 192-197, 2014. Article (CrossRef Link)
[32] D. Bienstock, M. Gonzalo, and Sebastian Pokutta, “Principled deep neural network training through linear programming,” *arXiv preprint arXiv:1810.03218*, 2018. Article (CrossRef Link)

Mehmet Sevri was born in Gaziantep, Turkey in 1987. He received the B.S. degree in Computer Engineering from Karadeniz Technical University, Trabzon in 2011 and M.S. degree in Information Systems from Gazi University, Ankara, in 2016. He is a Ph.D. candidate in Information System at Gazi University, Ankara.
He is a Research Assistant in the Informatics Institute at Gazi University since 2013. His research interest includes deep learning, cyber security and web application security.

Hacer Karacan was born in Erzurum, Turkey in 1980. She graduated from the department of computer education from Middle East Technical University (METU), Ankara in 2002. She received her master's degree in the cognitive science department, at METU in 2005, and completed her Ph.D. again in the cognitive science department, at METU in 2007. She started her academic career as a research assistant in the Department of Cognitive Science, METU in 2002. During her PhD studies, she worked as a visiting researcher at the University of Rochester (USA).
She is an Associate Professor in the Computer Engineering department at Gazi University, where she has been a faculty member since 2007. Her research focuses on using artificial intelligence methods to gain data insights. She is particularly interested in understanding data flow patterns and potential causal factors in the cyber security problems.