IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 18 May 2023, accepted 31 July 2023, date of publication 9 August 2023, date of current version 30 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3303845

RESEARCH ARTICLE

Web Based Anomaly Detection Using Zero-Shot Learning With CNN

DILEK YILMAZER DEMIREL AND MEHMET TAHIR SANDIKKAYA, (Senior Member, IEEE)
Department of Computer Engineering, Istanbul Technical University, 34469, İstanbul, Turkey
Corresponding author: Dilek Yilmazer Demirel (demirel18@itu.edu.tr)

ABSTRACT In recent years, attacks targeting websites have become a persistent threat. Therefore, web application security has become a significant issue. Dealing with unbalanced data is the biggest obstacle to providing security for web applications since there are fewer malicious requests despite a large number of benign requests. This paper suggests a novel Zero-Shot Learning method employing a Convolutional Neural Network (ZSL-CNN) to address unbalanced data problem and high false positive rates. This approach uses only benign data during training while predicting unseen malicious requests. Five web request datasets are used for validation on a diverse set of samples. The first dataset is a novel dataset containing Internet banking web request logs provided by Yapı Kredi Teknoloji. Other datasets are (i) an open-source WAF dataset, (ii) CSIC 2010 HTTP dataset, (iii) HTTP Params 2015 dataset, and (iv) a hybrid dataset. URIs are extracted from these datasets and fed to the ZSL-CNN model after code embedding. The same datasets are tested using other well-known models such as Isolation Forest, Autoencoder, Denoising Autoencoder with Dropout, and One-Class SVM. As per the comparison of the outcomes, it is seen that true positive rate of ZSL-CNN model is the greatest, reaching 99.29%.

INDEX TERMS Zero-shot learning, CNN, web attacks, attack detection, anomaly detection.

I. INTRODUCTION

Cybersecurity has grown in importance with the development of web-based applications, especially in light of the private and sensitive information that consumers and businesses around the world store. Companies use corporate networks to protect sensitive information, which is why it cannot be accessed directly from the internet. As a result, many attackers favor web application servers that act as end users' main points of contact [1], [2]. Data Breach Investigations Report [1] presents 56% of the violations are performed over web application servers [1], [2]. Additionally, figures on cybersecurity show that by 2025, the cost of global cybercrimes will reach $10.5 trillion per year [3]. Enterprise organizations have approximately 130 security breaches every year [3]. These results show that cybercrime is getting more costly every day [3]. Due to the growing focus and significance, numerous researchers have investigated the detection of cyberattacks using Machine Learning (ML) approaches including decision trees and support vector machines [4], [5]. However, ML approaches suffer from an impractical number of false-positive alarms.

The development of powerful computing devices makes easier training with Deep Learning (DL) models. DL techniques have accomplished successful results in many domains [6]. Especially in the security domain, these approaches have achieved less false alarm rates in studies for misuse detection and anomaly detection than ML approaches [7], [8]. Moreover, DL approaches are also utilized like a feature extractor for obtaining beneficial features using raw input such as text. In this manner, DL methods shorten the preprocessing and feature selection phase. Despite these advantages of DL methods, it emerges as a requirement that large volume of data must be fed into the model during training. Researchers use labeled data for supervised learning algorithms to obtain high classification accuracy. However, data is mostly not labeled in practice, so researchers suffer from mislabeled or less labeled data [9].

The associate editor coordinating the review of this manuscript and approving it for publication was Bing Li.

In the anomaly detection aspect, DL methods have difficulty producing results with high true positive rates, due to the fact that attacks are fewer detected. Additionally, existing methods cannot detect attacks in real time, as real-time attack detection consumes a lot of resources [9]. Moreover, there are also studies focusing on Web Application Firewall (WAF) usage with Machine Learning (ML), Features Engineering (FE), and DL techniques [2], [10]. WAFs aid in the protection of web applications by filtering and keeping track of HTTP traffic between a web application and the Internet. This paper presents a novel Zero-Shot Learning approach with CNN (ZSL-CNN) for web-based anomaly detection. This approach aims to handle zero-day web-based attacks and to obtain a high true-positive rate. In this manner, the impact of cybercrime could be decreased. The following list points out the major contributions:

- Providing a sample implementation design of the proposed model over a simplified realistic bank infrastructure, as shown in Section V.
- Presenting and implementing a novel approach utilizing Zero-Shot Learning with CNN for web-based attack detection, as given in Section VI.
- Building Isolation Forest [11], Autoencoder [12], Denoising Autoencoder with Dropout [13], and One-Class SVM [12] models as benchmark models in Section VII.
- Comparing the performance of the proposed model and benchmarks over five different data sets (a novel financial domain dataset, open-source WAF dataset [14], the CSIC 2010 dataset [15], HTTP Params 2015 dataset [16], and the hybrid dataset) in Section VIII.

The remainder of the paper is laid out as follows: The background of this study is given in Section II. Related works showing recent web-based attack detection studies are mentioned in Section III. Section IV defines the utilized datasets. Section VIII presents the experimental data from the proposed model and the benchmarks. Finally, the contributions are summarized in Section VIII-F.

II. BACKGROUND

In the background of this study, there are two basic concepts: Zero-Shot Learning and Convolutional-Neural Networks. This section presents the evolution of these concepts with mathematical representation.

A. ZERO-SHOT LEARNING

With the ever-increasing availability of information, it has become increasingly important to have a way to organize and classify data. One solution is the semantic representation or dataless classification. This solution appears first in the paper of Chang et al. [17]. This method uses natural language processing to group and categorizes data into meaningful clusters, making it easier to access and understand. Semantic representation allows data to be more efficiently and effectively organized, making it easier to identify patterns and make sense of the data. Then, Larochelle et al. give an introduction to Zero-data learning. They present an approach covering the cases with only descriptions of classes, and no training data [18]. Zero-Shot Learning (ZSL), which is inspired by human cognition capable of inferring results based on prior knowledge, is identified as a ML technique where it can learn from the data that has not appeared before training [19], [20]. Akin to people come across a new animal and classify it as a bird or cat, even if they've never seen that animal before [21].

The ZSL aims to simplify learning under extreme data imbalance. On the other hand, instances labeled with a subset of classes are used for training. At the same time, the auxiliary information is utilized to transfer knowledge for both unobserved and observed classes [19], [22], [23]. Akata et al. propose a label embedding approach to exploit the compatibility between images and their label embedding values [24]. In order to deal with different classes observed during training and test, *Lampert* et al. present a classification approach for attribute-based learning. In their study, semantic descriptions are used for learning with unobserved classes. A different strategy based on a semantic autoencoder rather than a projection function for ZSL is suggested by Kodirov et al. to deal with the domain drift problem [25].

In this study, a ZSL approach with CNN is implemented to detect anomalies, inspired by object recognition studies. For instance, Ba et al. presents a deep zero-shot convolutional neural network (ZS-CNN) that uses textual descriptions to predict image content accurately [26]. The proposed ZS-CNN model combines a convolutional neural network (CNN) with a deep recurrent neural network (RNN) to extract both visual and linguistic information from an image-text combination. Moreover, a brand-new contrastive loss function that maximizes the correlation between the predicted and target labels is utilized during the model training [26]. Unlike these studies, the proposed model uses a simple CNN and is used to detect anomalies in web requests.

B. CONVOLUTIONAL-NEURAL NETWORK (CNN)

Convolutional neural networks (CNNs) are identified as a type of neural network that has been used to great advantage in recognizing [27], classifying [28], and processing images and natural languages [29]. Like ordinary neural networks, CNNs are composed of several interconnected processing nodes or neurons that can recognize patterns in input data. Figure 1 [30] depicts a basic illustration of a neuron. A straightforward neuron can carry out the convolutional process by moving the kernel matrix across the input matrix. The process is repeated until all of the input is consumed as exemplified in Figure 1. The kernel and the overlapping data are multiplied, the results are accumulated, and finally, the bias error value is added on top of that [30].

These models are capable of extracting features from raw text by convolutional operations, without complex preprocessing. Text processing requires a focus on characters

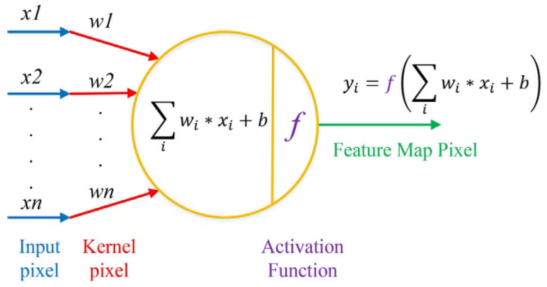

FIGURE 1. The simple neuron representation [30].

in a region rather than an isolated character. Therefore, local region features could be obtained for text classification [31]. Similarly, the features extracted from the URI are handled as a data stream using convolutional operations. This renders CNN a nice candidate for web-based anomaly detection.

Convolutional layers are made up of a series of learnable filters that are applied to the input data to generate feature maps. Each filter detects a certain characteristic in the input data. The input data is convolved with the filters. This process produces a feature map, which may be utilized as input for the next layer or as the output of the network. The following equation is used to obtain the feature map x_j:

$$x_j^l = \sigma\left(\sum_{i \epsilon M_j} x_i^{l-1} * k_{ij}^l + b_j\right) \tag{1}$$

where M_j shows the combination of the input feature maps, while k_{ij}^l represents a convolution kernel utilized to obtain the output j using the input i. The bias is given as b_j, and the activation function is represented as σ. The rectified linear unit (ReLU) activation function [32] is utilized as an activation function in the proposed model.

A max-pooling layer in a CNN performs a downsampling operation on the input. A max-pooling layer's function is to minimize the input's dimensionality so that the network can learn higher-order features. For example, if the input to a max-pooling layer is a 6×6 matrix, the max-pooling layer downsamples the input to a 3×3 matrix [30], [31]. The output k of the layer l is calculated below:

$$x_j^l = \max\left(x_{m*(k-1)+1}^{l-1}, \ldots, x_{m*(k-1)+n}^{l-1}\right) \tag{2}$$

A fully connected layer is a layer in a neural network where all nodes are connected to all nodes in the previous layer. This creates a many-to-many relationship between nodes.

Additionally, a normalization function is necessary for CNNs. The gradient could be any complex value, especially when the network is so deep. Batch normalization is a technique to improve the performance and stability of deep neural networks. It normalizes the activations of each neuron in a layer for each mini-batch. This decreases the likelihood of overfitting and speeds up the network's learning process [30], [31].

III. RELATED WORK

The majority of AI-based cybersecurity mechanisms require a considerable amount of labeled data for training. Labeled data is not always available. Zero-shot learning is a promising AI technique that can learn from only a few samples. It has been used effectively in a wide range of applications, including image classification and natural language processing.

The study of Rao and Mane suggests a zero-shot learning approach for an adaptive model that can learn from both labeled and unlabeled data. Additionally, the authors also develop an explainable AI approach that can provide insights into the learned models [11].

Zhang et al. propose a zero-shot learning approach to address the problem of detecting unknown attacks. This technique establishes a mapping from feature to semantic space that is used to identify unknown attacks by mapping the feature of known attacks to the semantic space, restoring the semantic space to the feature space by constraints of reconstruction error [23].

In order to overcome the difficulty of identifying unknown attacks with little or no training data, Komisarek et al., propose another ZSL approach. Their approach is based on the recent success of generative models in learning to generate realistic data samples. They train a generative model on a large dataset of normal network traffic flows. The trained model is then used to generate a synthetic dataset of regular flows [33].

On the other hand, deep learning-based approaches to anomaly detection also exist. For instance, Liu et al. propose Convolutional Neural Networks (CNNs) based and Recurrent Neural Networks (RNNs) based payload classification methods for attack detection. Their CNN-based method utilizes 1-dimensional CNN architecture with a max-pooling layer and a softmax layer, whereas their RNN-based method utilizes the Long Short-Term Memory (LSTM) architecture [31].

Tekerek et al. describe a novel CNN-based architecture for web-based threat detection. Current web-based attack detection systems frequently have limitations on the number of features they extract from web traffic. CNN helps to extract a variety of features from web traffic, including both static and dynamic features. The CNN is then able to classify web traffic as either benign or malicious [30].

Besides, many researchers focus on the application of autoencoders for anomaly detection [12], [13]. Mohamed et al. suggest a denoising autoencoder with dropout to boost anomaly detection's effectiveness. They also evaluate how well their strategy performs in comparison to other approaches like deep neural networks and support vector machines [13].

In addition to these studies, Toprak and Yavuz provide a DL-based web application firewall (WAF) to identify malicious requests. The WAF architecture is composed of two layers: an input preprocessing layer and an anomaly detection layer. Two modules make up the input preprocessing

layer: the request preprocessing module and the feature preprocessing module. The parsing of online requests and the extraction of their pertinent features are the responsibilities of the request preprocessing module. The retrieved features must be transformed by the feature preprocessing module into a format that is supplied to the anomaly detection layer. The Deep Neural Network (DNN)-based anomaly detection layer is taught to recognize malicious requests. The use of deep learning for WAF anomaly detection can result in more accurate and efficient detection of malicious activities, as compared to traditional methods [2].

Further, a WAF that makes use of machine learning and feature engineering to boost security against malicious traffic is described in another study. The WAF can recognize and prevent malicious requests after training on a dataset containing both benign and malicious requests. Machine learning techniques identify patterns and differentiate between malicious and benign requests, while feature engineering is used to create features from the raw data [10].

The comparison of the proposed approach and related studies is given in Table 1. The proposed approach presents several innovations and advances that differ from the existing literature on anomaly detection, as follows:

- First, the paper proposes a novel method for detecting anomalies in web-based data using zero-shot learning with CNN. This approach addresses the challenge of unbalanced data and focuses on decreasing the false-positive alarm rate. ZSL and CNN are applied for the first time in web-based anomaly detection. This approach uses an uncomplicated model to detect the majority of malicious queries and exposes significant improvements over recall compared to existing studies.
- Second, the paper applies the proposed method to real web-based data in the banking domain, which is a relatively unexplored area in the field of anomaly detection. The paper evaluates the effectiveness of the proposed method on different benchmark datasets, demonstrating its potential practical applicability.

IV. DATASETS

Web request logs are chronological records of all requests made to a web server. They can reveal a plethora of information about what anomalies have happened. Organizations could manually identify anomalies that signify malicious or unauthorized activity by monitoring and analyzing these logs. For instance, it might be a sign of malicious activity if a user suddenly starts continuous requests to a resource they have never used before [2]. This study utilizes five web request log datasets to evaluate the proposed model.

The first dataset utilized to evaluate the proposed model is a novel dataset provided by Yapı Kredi Technology (YKT). This dataset contains web request logs for inbound network traffic of an Internet banking site. This data consists of over 2 million benign requests and over 10 thousand malicious requests. URI is the sole part of input data utilized in the detection.

The second dataset [14] is an open-source dataset (WAF) containing over 1 million benign queries and over 40 thousand malicious queries that were collected from 30 different WAFs. It is designed to help researchers train machine learning models to detect and classify WAF attacks.

The third dataset [15] is the HTTP dataset CSIC 2010, which consists of thousands of web requests generated at the ''Information Security Institute'' of CSIC (Spanish Research National Council). This dataset contains more than 25 thousand malicious requests and 36 thousand normal requests. It is generated for use in testing web attack protection systems.

The fourth dataset is HTTP Params 2015 dataset. It is an open-source dataset that contains over 31 thousand instances. It consists of over 19 thousand benign queries and over 11 thousand malicious queries [16].

The fifth dataset is a hybrid dataset including benign queries in the banking dataset and malicious queries obtained from a firm named Picus Security. It contains over 2 million benign requests and over 5 thousand malicious requests.

Random sample sets containing benign and malicious instances are created with a 1:100 proportion for experimental results to tackle the unbalanced datasets. A few samples are given in Table 2.

V. USE CASE SCENERIOS

Many customers perform their transactions via Internet banking websites in the financial domain. Retail and corporate customers can check balances, create an account, transfer their money, apply for credit, and pay their bills. The sample uses cases for a banking customer are given in Fig 2. When a customer request is blocked because of a false positive alarm, it causes a loss of reputation and financial damage. On the other hand, banks have cyberattack detection systems based on strict rules to deal with many attack types. In order to resolve this contradiction, false positive rates on web-based attack detection need to be reduced.

Use Case Scenario 1: Initially, retail customer logs into the banking website. He checks his balances. Then, he browses the credit application page to apply for consumer credit. He types a string containing ''exe'' such as ''executive officer'' into the occupation input area on the credit application page. When the WAF rules have blocked all requests containing ''exe'', the application process interrupts. The proposed model evaluates the request under the anomaly detection module to get a more accurate prediction for anomalous requests. The anomaly detection module ensures that each request rather than the sequence of requests or the behavior of users is evaluated to determine whether it is malicious or not. Input data for this module is a GET request with query string parameters. In this manner, the customer request is blocked after it is checked to see if it is an anomaly. In a false positive alarm, the credit application cannot be completed, and a loss occurs since the credit cannot be sold.

TABLE 1. Comparison of the proposed approach and related studies.

Study	Year	ZSL	Method	Dataset
Rao et al. [11]	2021	✓	Isolation Forest	NSL-KDD
Zhang et al. [23]	2020	✓	Sparse Semantic Autoencoder	NSL-KDD
Komisarek et al. [33]	2022	✓	Random Forest	IoT-23 and SIMARGL2021
Liu et al. [31]	2019		CNN and RNN	DARPA1998
Tekerek et al. [30]	2021		CNN	CSIC2010v2
Mohamed et al. [13]	2019		Denoising AE with Dropout	NSL-KDD
Hindy et al. [12]	2020		Autoencoder	NSL-KDD and CICIDS2017
Toprak et al. [2]	2022		LSTM	CSIC 2010
Shaheed et al. [10]	2022		Machine Learning	CSIC 2010, HTTP Params 2015
Proposed Model	2023	✓	CNN	Banking Dataset, WAF Dataset, CSIC 2010, HTTP Params 2015, Hybrid Dataset

TABLE 2. Dataset samples.

Path	Dataset	Type
/javascript/nets.png	WAF Dataset	Benign
/javascript/info.exe	WAF Dataset	Malicious
/calendar/lang/calendar-tr.js	Banking Dataset	Benign
/ngi/index.do?lang=/etc/passwd	Banking Dataset	Malicious
/tienda1/publico/vaciar.jsp?B2A=Vaciar+carrito	CSIC 2010 Dataset	Benign
/scripts/tools/mkilog.exe	CSIC 2010 Dataset	Malicious
medioni5	HTTP Params 2015	Benign
-7387’))) order by 1–	HTTP Params 2015	Malicious
/ngi/eDevletMobile.do	Hybrid Dataset	Benign
/page975641.htm?p=select 9%2Cversion()	Hybrid Dataset	Malicious

Use Case Scenario 2: Corporate customer logs into the banking website. She checks credit cards and visualizes the credit card detail page. She wants to transfer money from a credit card to another account. She browses to the money transfer page and starts to fill in the relevant fields on the page. She types a special character in the description text area. When the WAF rules have blocked all requests containing special characters, money transfer transaction is interrupted. In order to avoid a false positive alarm, the operation is checked under the proposed anomaly detection model. This scenario uses a GET request with query string parameters as input to determine whether it is malicious or not. It focuses on each request independently and aims to reduce false positive rates. When a corporate customer transaction is interrupted, it might cause loss of corporate customers.

In general, a secure banking architecture consists of infrastructure components such as Intrusion Prevention System (IPS), Firewall (FW), Web Application Firewall (WAF), Load Balancer, and servers. The proposed approach is a candidate to be implemented as an anomaly detection extension of WAF. The banking application integration sample is shown in Figure 3. When a request is sent by the customer, this request can be evaluated by using an anomaly detection module, after the WAF rules detect it as a malicious. In this manner, false positives could be decreased.

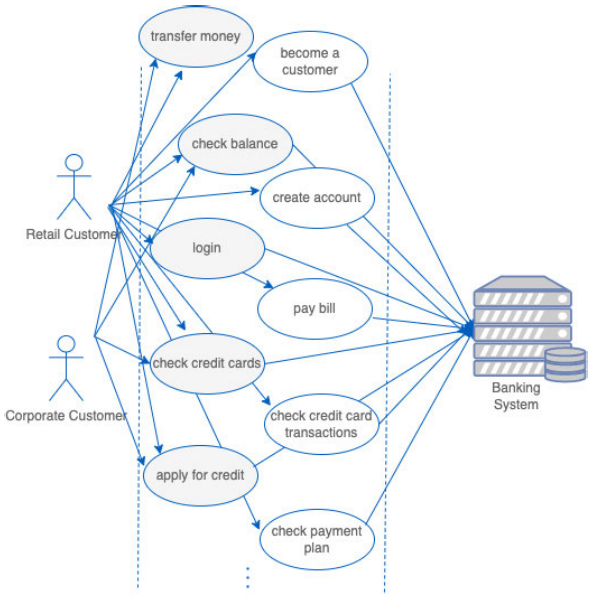

FIGURE 2. Banking System Use Cases: Use case scenarios consist of all operations performed by retail and corporate customers of a bank. The proposed model covers overall transactions in the banking domain. In this figure, only a few samples are presented.

VI. PROPOSED MODEL

This paper proposes a novel approach aiming to detect web-based anomalies. The proposed model (ZSL-CNN) is a Zero-Shot Learning model using CNN. It aims to increase true positive rates for web-based anomaly detection and, thus, reduce the operational expenditure of security while dealing with fewer anomalies. The proposed model trains on normal data, in other words, it learns how to map from input features to the desired labels. Therefore, seen classes are fed into the CNN model during training, then labels of unseen classes are predicted in the test phase.

The CNN architecture in the proposed model mainly consists of an input layer, two convolutional layers, two batch normalization layers, a max pooling layer, and a fully connected layer. The structure of this model is presented in

Figure 4. Additionally, the rectified linear unit (ReLU) activation function [32] is utilized as an activation function, and the Softmax function is used to calculate output values in the last layer [34]. The last layer is determined untrainable, therefore this layer does not affect the gradient updates during training. In this manner, it is aimed that the model learns how to establish the mapping between input data and labels.

The model mentioned above is performed in the training phase after benign data is fed into it. First, benign is labeled as "0" and is used as seen data. On the other hand, malicious data is used as unseen data in the test stage after being labeled as "1". The model gains the ability to map the desired labels by handling input data when training is completed. Second, the part of the trained model in the absence of the last layer, which contains the Softmax function, is taken for the test. After performing it with malicious data as unseen data, output vector values are obtained. Finally, the Euclidean distance is calculated between the output and label vectors. The label of an instance is identified according to the minimum distance. In other words, the minimum distance between label vectors and output vectors is indicative. The algorithm is given in Algorithm 1. The flowchart of the proposed model is depicted in Figure 5.

Algorithm 1 Web Based Anomaly Detection

Input: *BankingDataset*: Labeled dataset
$BankingDataset = ((x_1, y_1), \ldots, (x_{m+n}, y_{m+n}))$ with $m+n$ instances
X_{seen}: Benign data instances $X_{seen} = (x_1, \ldots, x_m)$ with m instances
Y_{seen}: Labels for benign data $Y_{seen} = (y_1, \ldots, y_m)$ with m instances
X_{unseen}: Malicious data instances $X_{unseen} = (x_1, \ldots, x_n)$ with n instances
Y_{unseen}: Labels for malicious data $Y_{unseen} = (y_1, \ldots, y_n)$ with n instances
Output: Y_{output}: Predicted labels for malicious data
TP: True positive instance count
FP: False positive instance count
Begin:
Initialize:
$X_{seen} \leftarrow [], Y_{seen} \leftarrow [], X_{unseen} \leftarrow [], Y_{unseen} \leftarrow []$

split Banking Dataset as *BenignQueries*, *MaliciousQueries*
$X_{seen}, Y_{seen} \leftarrow$ preprocess *BenignQueries* using code embedding
$X_{seen} \leftarrow MinMaxScaler(X_{seen})$
$X_{unseen}, Y_{unseen} \leftarrow$ preprocess *MaliciousQueries* using code embedding
$X_{unseen} \leftarrow MinMaxScaler(X_{unseen})$

train model with X_{seen}, Y_{seen} ;

$model_{zsl} \leftarrow$ remove the last layer from model
$class_{vectors} \leftarrow$ create label vectors using One-Hot Encoding
$pred_{vectors} \leftarrow$ predict vectors with $model_{zsl}$ using X_{unseen}

$Distances \leftarrow$ calculate Euclidean Distance between $pred_{vectors}$ and $class_{vectors}$
$Y_{output} \leftarrow$ get label list according to minimum distance
$TP, FP \leftarrow$ calculate *TP* and *FP* using Y_{output}, Y_{unseen}

End of algorithm

A. DATA PREPROCESSING

In this study, code embedding is used as data preprocessing [35]. Code embedding is a process where characters are represented as vectors by using ASCII codes. This allows for characters to be compared and clustered based on their similarity in the vector space. This can be useful for tasks such as language modeling and machine translation. Code embeddings have several advantages over one-hot encodings. First, they are much smaller in size. A one-hot encoding requires a vector of size N (where N is the number of ASCII values), whereas a code embedding only requires a matrix of size M × N (where M is the size of the character embedding). *Jemal* et al. represent that code embeddings have been shown to improve the performance of CNNs on malicious HTTP request detection [35].

The Uniform Resource Identifier (URI) path is used as input data in this work. It is the tail of the URI string after the host name. Code embedding is carried out on this data to represent them as vectors. Each input is broken down into characters and given an integer value using ASCII code values while executing code embedding. Table 3 displays the use of ASCII code values.

B. NORMALIZATION

The existence of instances with various feature dimensions causes significant problems such as slowness while training, and minor changes for accuracy increase. The MinMaxScaler is used in this study [36] to deal with these problems. The MinMaxScaler scales the data features in the range of 0 to 1. The equation for the MinMaxScaler is shown below:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3}$$

where x_{max} represents the maximum value between data and x_{min} represents the minimum value between data [36].

C. EVALUATION METRICS

This study focuses on recall as an evaluation metric, because the test process is performed with only malicious data. In this case, if a malicious request (Actual Positive) is predicted as benign (Predicted Negative), the consequence of this result is very costly for the service provider. As a result, recall is an inductive metric for this study. The recall is the number of true positives divided by the sum of the true positives and the false negatives. It is calculated by the following formula [37].

$$Recall(TruePositiveRate) = \frac{TP}{TP + FN} \tag{4}$$

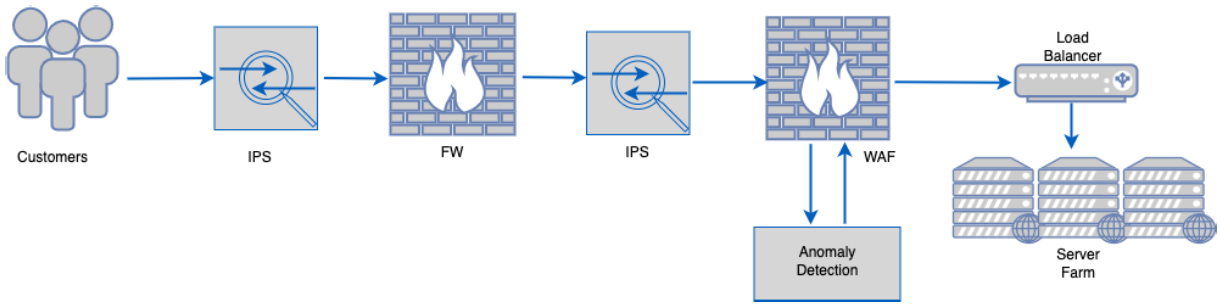

FIGURE 3. Banking Application Integration Sample: A simplified realistic bank infrastructure consists of Intrusion Prevention Systems (IPS), Firewall (FW), Web Application Firewall (WAF), Load Balancer, and servers. Considering this architecture, the anomaly detection module is implemented as an extension of WAF.

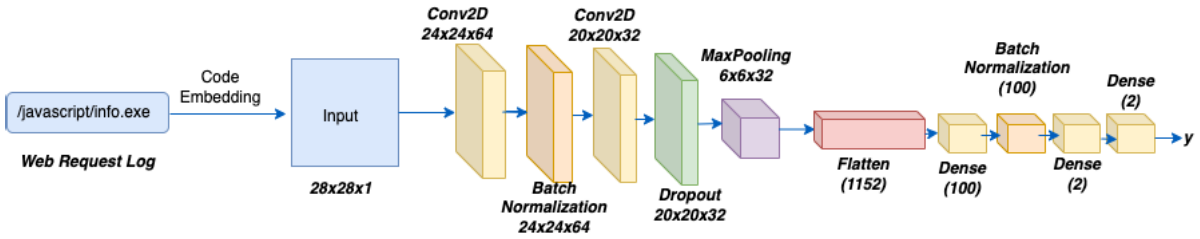

FIGURE 4. The CNN Architecture in the proposed model: The path obtained from the web request logs is given as input. Then, the input is processed by convolutional, batch normalization, max-pooling, and fully-connected layers.

VII. BENCHMARK MODELS

Aiming the evaluation of the proposed approach, the benchmark models are implemented. Trials are performed with the same datasets. SVM and Naive Bayes algorithms do not predict any anomaly instances after training only benign data. So that, Isolation Forest, One Class SVM, Autoencoder, and Denoising Autoencoder are used as benchmarks. This section will explain how these models are implemented.

First, the Isolation Forest algorithm utilized in the study of Rao and Mane [11] is established. Rao and Mane [11] presents a ZSL approach for adaptive cybersecurity that can learn from both labeled and unlabeled data. In the case of anomaly detection, they train their model with normal network traffic data. Then the test is performed by using only anomalies. To compare with the proposed model, a similar technique is used over the Isolation Forest algorithm. The implementation of this algorithm is presented in Algorithm 2.

An isolation forest is a type of machine-learning algorithm that is used to identify anomalies. It works by randomly selecting a feature of the data and then splitting the data on that feature. The algorithm then repeats this process until all the data points have been split. The anomaly score for a data point is the number of times that the data point was the only point in a split. The more times a data point is the only point in a split, the more likely it is to be an anomaly [11].

In the study of Hindy et al. [12], an autoencoder implementation is proposed to detect zero-day attacks. This study aims to establish an IDS model that has both high recall and keeps miss rate (false-negatives) to an acceptable minimum. It is performed over well-known IDS datasets. For comparison efficiency, they use a one-class support vector machine (SVM) method and achieve a high degree of accuracy in identifying zero-day attacks, by taking advantage of autoencoder encoding/decoding capabilities. Considering this study, autoencoder and One-Class SVM implementations are performed as benchmarks.

An autoencoder aims to transform the input into a hidden representation that contains the most important information from the input. The hidden representation is then decoded back into the original input. Therefore, it can obtain a threshold to separate classes [12]. The usage of an autoencoder for web attack detection is presented in Algorithm 3. *CalculateReconstructionError* and *PredictClass* functions are given in Appendix A. In the implementation of the autoencoder, only benign data is given as input during training. Then, the reconstruction error values are calculated using benign and anomaly data. By using these construction errors, a threshold value is determined. Finally, the threshold value is utilized to predict the class.

The threshold value is calculated as approximately 0.0157. During the evaluation of the WAF dataset, the alpha value is used as 1.75 to deal with the overfitting. Additionally, the threshold value is calculated as approximately 0.0194. For CSIC 2010 dataset, the alpha value is again utilized as 1.75, and the threshold value is obtained as approximately 0.05732. The alpha value for HTTP Params 2015 is 1.75 and the threshold is 0.0156. Finally, the alpha value for the hybrid dataset is used as 0.5 and the threshold is calculated as s0.0150.

TABLE 3. ASCII code samples.

Path	ASCII Code Value
/javascript/nets.png	[47, 106, 97, 118, 97, 115, 99, 114, 105, 112, 116, 47, 110, 101, 116, 115, 46, 112, 110, 103]
/ref010203/	[47, 114, 101, 102, 48, 49, 48, 50, 48, 51, 47]

Algorithm 2 Benchmark 1 - Isolation Forest

Input: *BankingDataset*: Labeled dataset $BankingDataset = ((x_1, y_1), \ldots, (x_{m+n}, y_{m+n}))$ with $m + n$ instances
X_{seen}: Benign data instances $X_{seen} = (x_1, \ldots, x_m)$ with m instances
Y_{seen}: Labels for benign data $Y_{seen} = (y_1, \ldots, y_m)$ with m instances
X_{unseen}: Malicious data instances $X_{unseen} = (x_1, \ldots, x_n)$ with n instances
Y_{unseen}: Labels for malicious data $Y_{unseen} = (y_1, \ldots, y_n)$ with n instances
Output: Y_{output}: Predicted labels for malicious data
TP: True positive instance count
FP: False positive instance count
Begin:
Initialize: $X_{seen} \leftarrow []$, $Y_{seen} \leftarrow []$, $X_{unseen} \leftarrow []$, $Y_{unseen} \leftarrow []$
split Banking Dataset as *BenignQueries*, *MaliciousQueries*
X_{seen}, $Y_{seen} \leftarrow$ preprocess *BenignQueries* using code embedding
$X_{seen} \leftarrow MinMaxScaler(X_{seen})$
X_{unseen}, $Y_{unseen} \leftarrow$ preprocess *MaliciousQueries* using code embedding
$X_{unseen} \leftarrow MinMaxScaler(X_{unseen})$
train Isolation Forest model with X_{seen}, Y_{seen} ;
$Y_{output} \leftarrow$ predict labels with Isolation Forest model using X_{unseen}
TP, *FP* $\leftarrow$ calculate *TP* and *FP* using Y_{output}, Y_{unseen}
End of algorithm

Algorithm 3 Benchmark 2 - Autoencoder (AE)

Input: *BankingDataset*: Labeled dataset $BankingDataset = ((x_1, y_1), \ldots, (x_{m+n}, y_{m+n}))$ with $m + n$ instances
X_{seen}: Benign data instances $X_{seen} = (x_1, \ldots, x_m)$ with m instances
Y_{seen}: Labels for benign data $Y_{seen} = (y_1, \ldots, y_m)$ with m instances
X_{unseen}: Malicious data instances $X_{unseen} = (x_1, \ldots, x_n)$ with n instances
Y_{unseen}: Labels for malicious data $Y_{unseen} = (y_1, \ldots, y_n)$ with n instances
Output: Y_{output}: Predicted labels for malicious data
TP: True positive instance count
FP: False positive instance count
Begin:
Initialize: $X_{seen} \leftarrow []$, $Y_{seen} \leftarrow []$, $X_{unseen} \leftarrow []$, $Y_{unseen} \leftarrow []$
split Banking Dataset as *BenignQueries*, *MaliciousQueries*
X_{seen}, $Y_{seen} \leftarrow$ preprocess *BenignQueries* using code embedding
$X_{seen} \leftarrow MinMaxScaler(X_{seen})$
X_{unseen}, $Y_{unseen} \leftarrow$ preprocess *MaliciousQueries* using code embedding
$X_{unseen} \leftarrow MinMaxScaler(X_{unseen})$
train Autoencoder model with X_{seen}, Y_{seen}
$re_{seen} \leftarrow CalculateReconstructionError(X_{seen}, model)$
$re_{unseen} \leftarrow CalculateReconstructionError(X_{unseen}, model)$
$alpha \leftarrow 0.5$
threshold $\leftarrow$ multiply *alpha* and mean of the concatenation of re_{seen} and re_{unseen}
$Y_{output} \leftarrow$ predict labels using $X_{anomaly}$ and *threshold*;
TP, *FP* $\leftarrow$ calculate *TP* and *FP* using Y_{output}, Y_{unseen}
End of algorithm

The second benchmark model is One-class SVM which is a machine learning algorithm that is used for anomaly detection. It is an unsupervised learning algorithm that is trained on a dataset with only one class. The algorithm creates a decision boundary that separates the data points in the training set from the rest of the data. The decision boundary is created by maximizing the margin between the data points and the boundary. The algorithm is then able to detect new data points that are not part of the training set by seeing which side of the decision boundary they fall on. If the new data point falls on the same side of the boundary as the training data, then it is considered to be part of the same class. If it falls on the other side of the boundary, then it is considered to be an anomaly [12]. The usage of One-class SVM for web-based attack detection is presented in Algorithm 5. Although it is an unsupervised learning technique, it is utilized to gain

Algorithm 4 Benchmark 4 - Denoising Autoencoder With Dropout (DAE)

Input: *BankingDataset*: Labeled dataset $BankingDataset = ((x_1, y_1), \ldots, (x_{m+n}, y_{m+n}))$ with $m + n$ instances
X_{seen}: Benign data instances $X_{seen} = (x_1, \ldots, x_m)$ with m instances
Y_{seen}: Labels for benign data $Y_{seen} = (y_1, \ldots, y_m)$ with m instances
X_{unseen}: Malicious data instances $X_{unseen} = (x_1, \ldots, x_n)$ with n instances
Y_{unseen}: Labels for malicious data $Y_{unseen} = (y_1, \ldots, y_n)$ with n instances
Output: Y_{output}: Predicted labels for malicious data
TP: True positive instance count
FP: False positive instance count
Begin:
Initialize: $X_{seen} \leftarrow []$, $Y_{seen} \leftarrow []$, $X_{unseen} \leftarrow []$, $Y_{unseen} \leftarrow []$
split Banking Dataset as *BenignQueries*, *MaliciousQueries*
X_{seen}, $Y_{seen} \leftarrow$ preprocess *BenignQueries* using code embedding
$X_{seen} \leftarrow MinMaxScaler(X_{seen})$
X_{unseen}, $Y_{unseen} \leftarrow$ preprocess *MaliciousQueries* using code embedding
$X_{unseen} \leftarrow MinMaxScaler(X_{unseen})$
train Denoising Autoencoder model with X_{seen}
threshold $\leftarrow$ get the last loss value from the model history
pred $\leftarrow$ predict labels using model and X_{unseen}
$loss_{test} \leftarrow$ calculate losses using *pred* and X_{unseen}
$Y_{output} \leftarrow$ create a zero array with the length of malicious data
for each element of array Y_{output}, determine it as malicious if it exceeds the threshold
TP, *FP* $\leftarrow$ calculate *TP* and *FP* using Y_{output}, Y_{unseen}

End of algorithm

Algorithm 5 Benchmark 3 - One-Class SVM

Input: *BankingDataset*: Labeled dataset $BankingDataset = ((x_1, y_1), \ldots, (x_{m+n}, y_{m+n}))$ with $m + n$ instances
X_{seen}: Benign data instances $X_{seen} = (x_1, \ldots, x_m)$ with m instances
Y_{seen}: Labels for benign data $Y_{seen} = (y_1, \ldots, y_m)$ with m instances
X_{unseen}: Malicious data instances $X_{unseen} = (x_1, \ldots, x_n)$ with n instances
Y_{unseen}: Labels for malicious data $Y_{unseen} = (y_1, \ldots, y_n)$ with n instances
Output: Y_{output}: Predicted labels for malicious data
TP: True positive instance count
FP: False positive instance count
Begin:
Initialize: $X_{seen} \leftarrow []$, $Y_{seen} \leftarrow []$, $X_{unseen} \leftarrow []$, $Y_{unseen} \leftarrow []$
split Banking Dataset as *BenignQueries*, *MaliciousQueries*
X_{seen}, $Y_{seen} \leftarrow$ preprocess *BenignQueries* using code embedding
$X_{seen} \leftarrow MinMaxScaler(X_{seen})$
X_{unseen}, $Y_{unseen} \leftarrow$ preprocess *MaliciousQueries* using code embedding
$X_{unseen} \leftarrow MinMaxScaler(X_{unseen})$
$X \leftarrow$ contanate X_{seen} and X_{unseen}
$Y \leftarrow$ contanate Y_{seen} and Y_{unseen}
train One-Class SVM model with X
pred $\leftarrow$ predict labels with One-Class SVM model using X
$Y_{output} \leftarrow$ get labels for only malicious data using *pred*
TP, *FP* $\leftarrow$ calculate *TP* and *FP* using Y_{output}, Y_{unseen}
End of algorithm

more insight with respect to classic supervised ML methods such as SVM, and Naive Bayes. The hyperparameters are used akin to the given algorithm for each dataset. In this implementation, both anomaly and benign data are used as input. Then, a decision boundary is extracted to determine outputs either as anomalous or benign. Only anomaly data is taken into account for comparison with the proposed model.

Mohamed et al. [13] propose to use a dropout-based autoencoder for network anomaly detection. The autoencoder is trained using normal traffic data, and the dropout is used to prevent overfitting. The autoencoder is then used to reconstruct new traffic data, and the reconstruction error is used to detect anomalies. They evaluate their method on a real-world network traffic dataset and show that their method outperforms existing methods. In this approach, a threshold value is utilized to set a label for each instance; hence, this approach is implemented with a similar threshold technique. The thresholds are calculated as approximately 0.0026 for the banking dataset and 0.0017 for the WAF dataset. Additionally, the threshold is approximately calculated as 0.0048 for the CSIC 2010 dataset. Threshold values for HTTP Params 2015 and

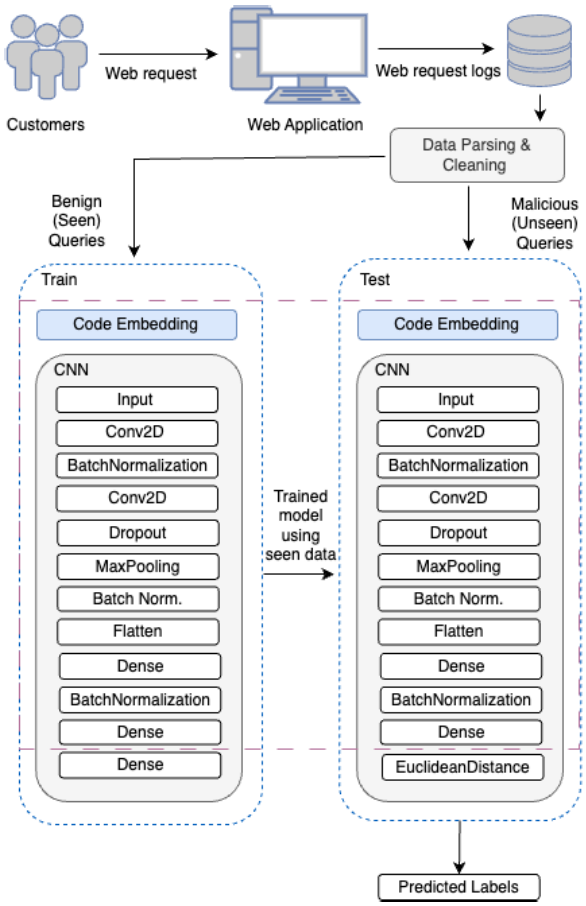

FIGURE 5. The flowchart of the proposed model: Users make requests to the web application. The web requests are investigated to label malicious and benign queries. Next, the proposed model is trained using benign data. The part of the model other than the last layer is used for testing.

the hybrid datasets are calculated as 0.0016 and 0.0053, respectively.

The autoencoder pseudocode is given in Algorithm 4. *CalculateLosses* function is given in Appendix B. First, only benign data is used during training. Then, the threshold value is determined using the loss value after the model is trained. According to this threshold, output classes are predicted.

VIII. RESULTS

The experiments are performed on a machine with 16 GB of RAM and 64 GB of swap space with AMD Ryzen 7 2700X CPU and NVIDIA GTX 1060 GPU. Further, Tensorflow 2.10[1] and Python 3.8[2] are used to conduct the implementations.

A. EVALUATION WITH BANKING DATA

The random sample sets are prepared with 1:100 proportions (141 anomaly instances and 14100 benign instances) to evaluate the proposed model in the banking domain. The preparation is repeated five times to clarify effectiveness. The proposed approach is performed, and better results (0.9929) are obtained than the benchmark models using the banking data set. The result of the Isolation Forest algorithm is measured as 0.4397 as the maximum value. The recall rates of Autoencoder and Denoising Autoencoder are respectively 0.8085 and 0.8723, after tuning the threshold values. Furthermore, the recall rate of the One-Class SVM is 0.6312. Considering these results, the proposed model achieved practical effectiveness. These results are given in Figure 6.

B. EVALUATION WITH OPEN-SOURCE WAF DATA

An open-source WAF data set is used to verify this approach in a different domain [14]. The random sample sets are prepared with 1:100 proportions (141 anomaly instances and 14100 benign instances) to have the same condition. This process is again repeated five times.

The proposed model's evaluation indicated better results (0.9929) than the benchmark models did. The Isolation Forest technique yields a maximum value of 0.8936, and after adjusting the threshold values, the recall rates of the Autoencoder and Denoising Autoencoder are respectively 0.9219 and 0.9503. Additionally, One-Class SVM's recall rate is 0.4326. In light of these findings, the suggested model is very promising. Figure 7 presents these findings.

C. EVALUATION WITH THE HTTP DATASET CSIC 2010

The HTTP Dataset CSIC 2010 [15] is utilized to verify this approach in e-commerce web applications. The random sample sets are prepared with 1:100 proportions (141 anomaly instances and 14100 benign instances) to mimic the same condition. This process is repeated five times.

The proposed model again outperforms (0.9929) other models for CSIC 2010 dataset. The Isolation Forest approach provides a maximum value of 0.8723, and after changing the threshold values, the Autoencoder and Denoising Autoencoder recall rates are 0.7375 and 0.9645, respectively. Furthermore, the recall rate of the One-Class SVM is 0.6099. The measurements are depicted in Figure 8.

D. EVALUATION WITH THE HTTP PARAMS 2015

The HTTP Params 2015 dataset [16] is experimented with the same configurations as before. Random sample sets are generated and the experiment is repeated five times.

The proposed approach fared better than the benchmark models (0.9929). After adjusting the threshold settings, the Autoencoder recall rate is 0.8865, and both the Isolation Forest and Denoising Autoencoder approaches offer a maximum value of 0.9858. Additionally, the One-Class SVM's recall rate is 0.6950. Figure 9 visualizes the measurements.

[1]TensorFlow Library. Available online: https://www.tensorflow.org
[2]Python Software Foundation. Available online: https://www.python.org/

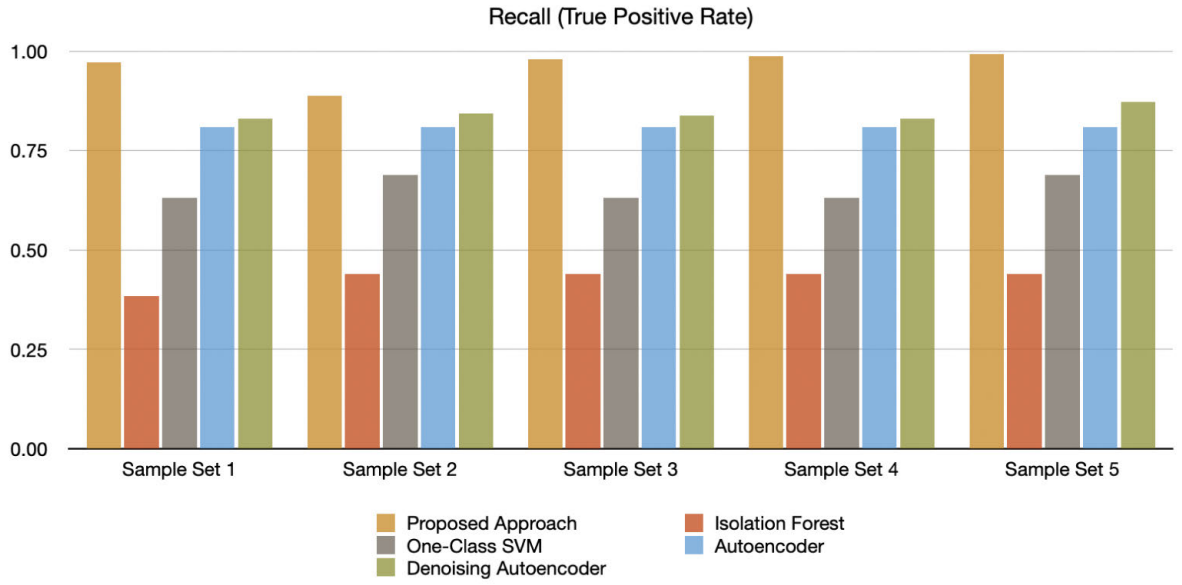

FIGURE 6. Recall (True Positive Rates) for Banking Data: Five random sample datasets created from Banking Data are used for comparison with benchmarks and the proposed model.

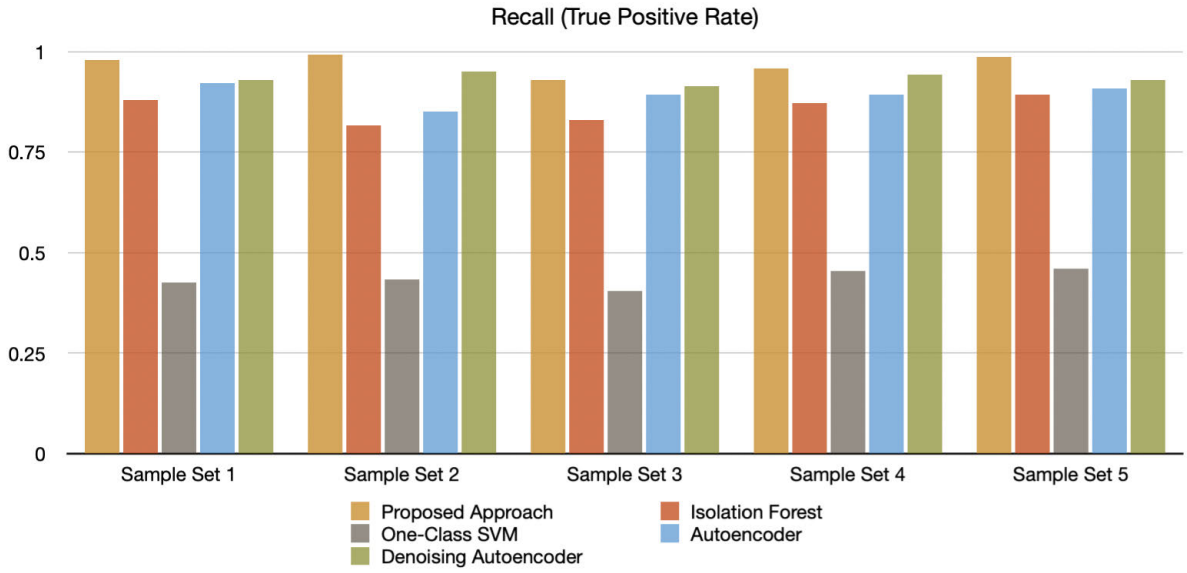

FIGURE 7. Recall (True Positive Rates) for WAF Data: Five random sample datasets created from WAF Data are used for comparison with benchmarks and the proposed model.

E. EVALUATION WITH THE HYBRID DATA

Effectiveness of the proposed model is put on trial by an additional dataset. Hybrid dataset contains several attack requests provided by an active security firm. Identical conditions are built by the random sample sets and five times of repetition.

The proposed model outperforms the benchmark models one more time (0.9716). The Autoencoder and Denoising Autoencoder recall rates are 0.8581 and 0.6737, respectively. The Isolation Forest approach provides a maximum value of 0.4255. Furthermore, the One-Class SVM has a recall rate of 0.8014. Figure 10 depicts experimental results.

F. SUMMARY

The best results of the recall metric are shown in Table 4. Isolation Forest and Autoencoder have higher results for the WAF dataset than the results for the banking and CSIC 2010 datasets. In addition, Isolation Forest is effective for HTTP Params 2015. Denoising Autoencoder has a higher recall value for CSIC 2010 dataset. HTTP Params

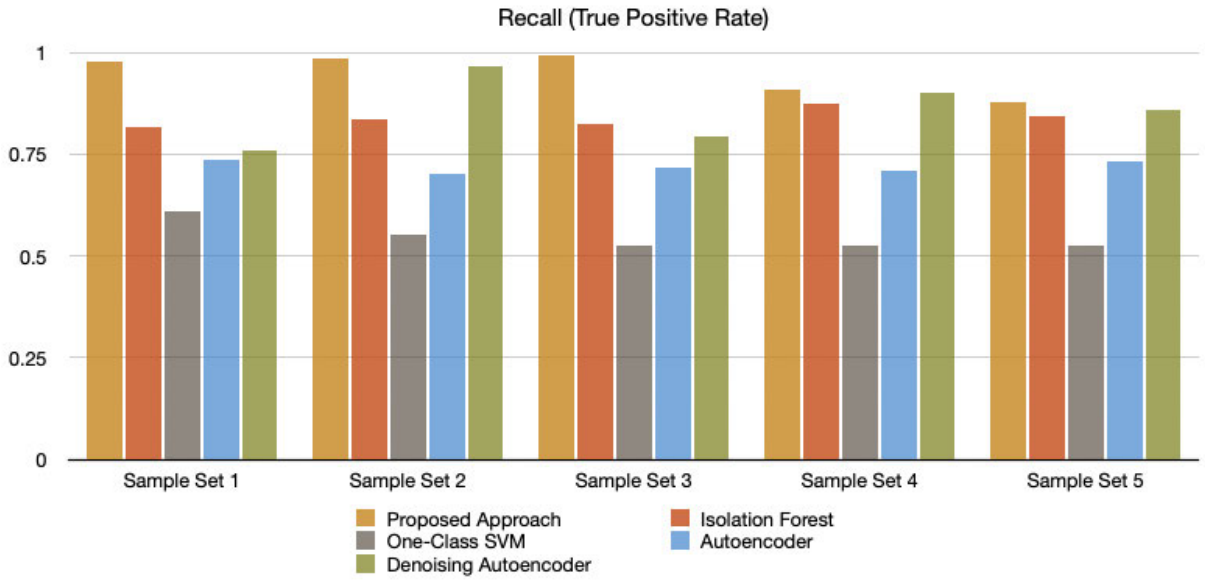

FIGURE 8. Recall (True Positive Rates) for CSIC 2010 Data: Five random sample datasets created from CSIC 2010 Data are used for comparison with benchmarks and the proposed model.

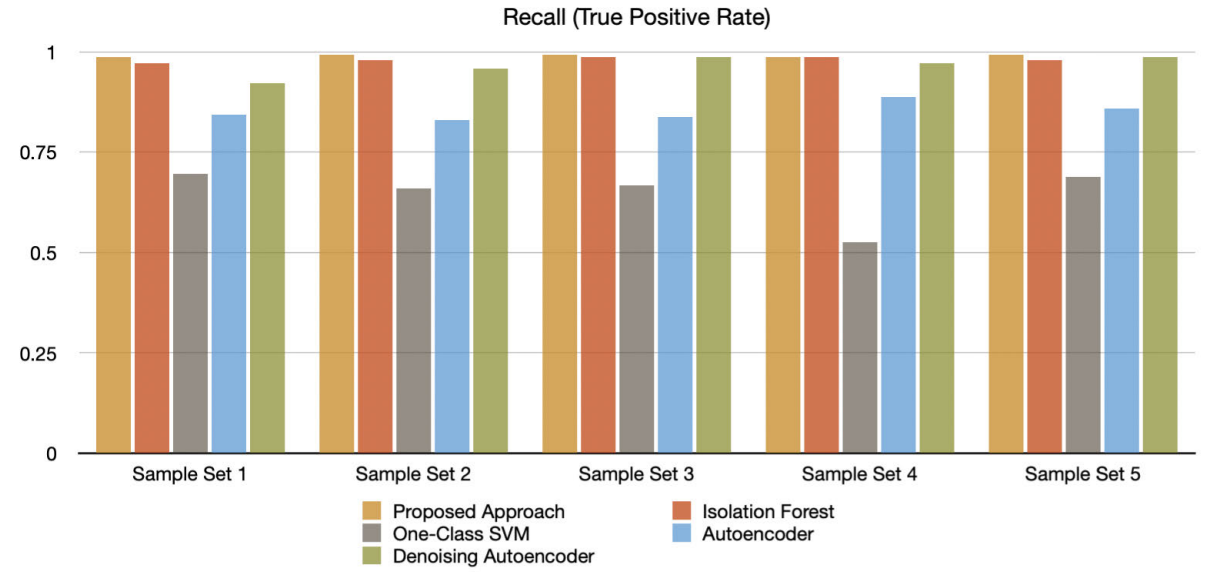

FIGURE 9. Recall (True Positive Rates) for HTTP Params 2015 Data: Five random sample datasets created from HTTP Params 2015 Data are used for comparison with benchmarks and the proposed model.

2015 dataset has higher values than values for other datasets. One-Class SVM achieves a better recall value for the Hybrid dataset than recall values for other datasets. The Isolation Forest has the lowest recall rate for the banking dataset according to all test results, while One-Class SVM has the lowest recall value for the WAF dataset. Autoencoder and Denoising Autoencoder models usually show good performance for all datasets. In addition to these results, the proposed model has demonstrated the best performance for all datasets compared to the benchmark models. The recall of the proposed model is 12% better than the recall of the Denoising Autoencoder for banking data, 4% for WAF data, 3% for CSIC 2010 data, 1% for HTTP Params 2015 data, and 30% for Hybrid data. The proposed model outperforms the Isolation Forest for banking data by more than 2x, WAF data by 10%, CSIC 2010 data by 12%, HTTP Params 2015 data by 1%, and Hybrid data by 55%. The recall of the proposed model exceeds the One-Class SVM by 31% for banking data, more than 2x for WAF data, 39% for CSIC 2010 data, 30% for HTTP Params 2015 data, and 17% for Hybrid data. Additionally, the proposed model gets ahead of the Autoencoder by 19% for banking data, 7% for WAF data, 26% for CSIC 2010 data, 11% for HTTP Params 2015 data, and 12% for Hybrid data.

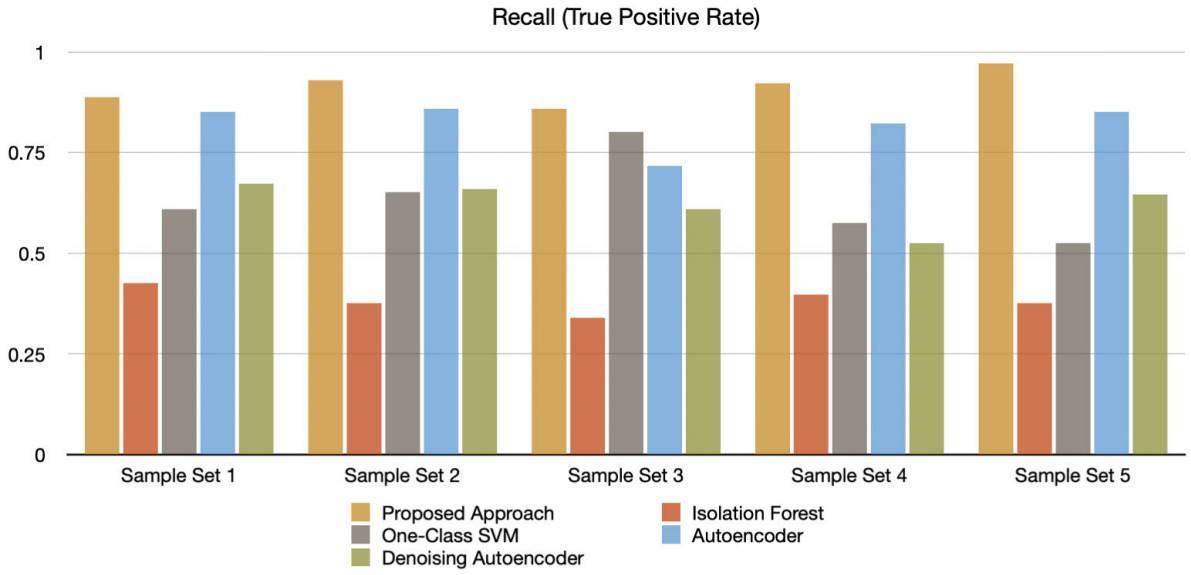

FIGURE 10. Recall (True Positive Rates) for Hybrid Data: Five random sample datasets created from Hybrid Data are used for comparison with benchmarks and the proposed model.

TABLE 4. Best results (recall).

Method	Banking Data	WAF Data	CSIC2010 Data	HTTP Params 2015 Data	Hybrid Data
Proposed Approach	0.9929	0.9929	0.9929	0.9929	0.9716
Isolation Forest	0.4397	0.8936	0.8723	0.9858	0.4255
One-Class SVM	0.6879	0.4609	0.6099	0.6950	0.8014
Autoencoder	0.8085	0.9219	0.7375	0.8865	0.8581
Denoising Autoencoder	0.8723	0.9503	0.9645	0.9858	0.6737

IX. DISCUSSION

The proposed model achieves high true positive rate for anomaly detection in the presence of unbalanced data. However, the model would be better updated by training actual normal data to avoid zero days. Still, the model could be improved with the Generalized Zero-Shot Learning methodology to be useful for both malicious and benign requests after being integrated as a WAF extension.

Furthermore, an adaptive approach consisting of two phases might help. The first phase could be a classification module to classify a request as malicious or benign. The second phase could be a ZSL module to determine the type of attacks.

The proposed model is currently capable of detecting malicious attacks. Further, it could be improved to classify attack types. Dataset diversity might be helpful for the enriched set of attack types.

Algorithm 6 Recontruction Error Calculation for AE

Input: X, $model$;
Output: $error$;
$r \leftarrow model.predict(X)$;
return $MeanSquaredError(X, r)$;

X. CONCLUSION

This paper proposes a novel web-based anomaly detection approach using zero-shot learning with CNN. According to experiments, the model has demonstrated its effectiveness in detecting anomalies on five distinct datasets. This approach has the potential to be further improved with other machine-learning techniques. It is anticipated that ZSL-CNN serves as a security-enhancing tool to be integrated with existing WAFs for detecting anomalies, especially in financial web-based applications.

APPENDIX A ALGORITHMS

The functions called in Algorithm 3 (the implementation of Autoencoder) are presented in this section.

CalculateReconstructionError function is given in Algorithm 6.

PredictClass function is given in Algorithm 7.

The functions used in Algorithm 4 (the implementation of Denoising Autoencoder with Dropout) are presented in this section.

CalculateLosses function is given Algorithm 8.

Algorithm 7 Class Prediction for AE

Input: X, $threshold$, $model$;
Output: $label$;
$re_{error} \leftarrow CalculateReconstructionError(X, model)$;
if $re_{error} \geq threshold$ **then**
  return 0;
else
  return 1;
end

Algorithm 8 Loss Calculation for DAE

Input: X, $preds$;
Output: $losses$;
Initialize: $losses \leftarrow zeros(len(x))$;

for $i \leftarrow 0$ **to** $len(x)$ **do**
  $losses[i] \leftarrow sqrt(preds[i] - x[i]).mean()$
end

return $losses$;

APPENDIX B
ABBREVIATIONS

The abbreviations used in this manuscript are shown in Table 5.

TABLE 5. Abbreviations.

Abbreviation	Definition
ML	Machine Learning
FE	Feature Engineering
ZSL	Zero-Shot Learning
DL	Deep Learning
DAP	Direct Attribute Prediction
ALE	Attribute Label Embedding
CNN	Convolutional-Neural Network
URI	Uniform Resource Identifier
LSTM	Long Short-Term Memory
ReLU	Rectified Linear Unit
AI	Artificial Intelligence
SVM	Support Vector Machine
FN	False Negative
TPR	True Positive Rate
TP	True Positive
YKT	Yapı Kredi Technology
AE	Autoencoder
DAE	Denoising Autoencoder with Dropout

REFERENCES

[1] S. Mansfield-Devine, "Verizon: Data breach investigations report," *Comput. Fraud Secur.*, vol. 2022, no. 6, pp. 1–72, Jun. 2022.
[2] S. Toprak and A. Yavuz, "Web application firewall based on anomaly detection using deep learning," in *Proc. ACTA INFOLOGICA*, 2022, pp. 142–149.
[3] PurpleSec. *Cyber Security Statistics: The Ultimate List of Stats Data, & Trends for 2022*. Accessed: Dec. 5, 2022. [Online]. Available: https://web.archive.org/web/20221205155455/ and https://purplesec.us/resources/cyber-security-statistics/
[4] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines," in *Proc. Int. Joint Conf. Neural Networks*, 2002, pp. 1702–1707.
[5] G. Stein, B. Chen, A. S. Wu, and K. A. Hua, "Decision tree classifier for network intrusion detection with GA-based feature selection," in *Proc. 43rd Annu. Southeast regional Conf. Volume 2*, Mar. 2005, pp. 136–141.
[6] J. Liang, W. Zhao, and W. Ye, "Anomaly-based web attack detection: A deep learning approach," in *Proc. VI Int. Conf. Netw., Commun. Comput.*, Dec. 2017, pp. 80–85.
[7] T.-T.-H. Le, J. Kim, and H. Kim, "An effective intrusion detection classifier using long short-term memory with gradient descent optimization," in *Proc. Int. Conf. Platform Technol. Service (PlatCon)*, Feb. 2017, pp. 1–6.
[8] A. M. Vartouni, S. S. Kashi, and M. Teshnehlab, "An anomaly detection method to detect web attacks using stacked auto-encoder," in *Proc. 6th Iranian Joint Congr. Fuzzy Intell. Syst. (CFIS)*, Feb. 2018, pp. 131–134.
[9] X. Gong, J. Lu, Y. Zhou, H. Qiu, and R. He, "Model uncertainty based annotation error fixing for web attack detection," *J. Signal Process. Syst.*, vol. 93, nos. 2–3, pp. 187–199, Mar. 2021.
[10] A. Shaheed and M. H. D. B. Kurdy, "Web application firewall using machine learning and features engineering," *Secur. Commun. Netw.*, vol. 2022, pp. 1–14, Jun. 2022.
[11] D. Rao and S. Mane, "Zero-shot learning approach to adaptive cybersecurity using explainable AI," 2021, *arXiv:2106.14647*.
[12] H. Hindy, R. Atkinson, C. Tachtatzis, J.-N. Colin, E. Bayne, and X. Bellekens, "Utilising deep learning techniques for effective zero-day attack detection," *Electronics*, vol. 9, no. 10, p. 1684, Oct. 2020.
[13] S. Mohamed, R. Ejbali, and M. Zaied, "Denoising autoencoder with dropout based network anomaly detection," in *Proc. ICSEA*, 2019, pp. 1–10.
[14] F. Ahmad. (2017). *WAF Malicious Queries Data Sets*. Accessed: Mar. 1, 2023. [Online]. Available: https://web.archive.org/web/20230301151428/https://github.com/faizann24/Fwaf-Machine-Learning-driven-Web-Application-Firewall/
[15] I S Institute. (2012). *HTTP DATASET CSIC 2010*. Accessed: Dec. 18, 2014. [Online]. Available: https://www.tic.itefi.csic.es/dataset/
[16] (2015). *HTTP Params Dataset*. Accessed: Aug. 3, 2023. [Online]. Available: https://web.archive.org/web/20230803085051/ and https://github.com/Morzeux/HttpParamsDataset
[17] M.-W. Chang, L.-A. Ratinov, D. Roth, and V. Srikumar, "Importance of semantic representation: Dataless classification," in *Proc. AAAI*, vol. 2, 2008, pp. 830–835.
[18] H. Larochelle, D. Erhan, and Y. Bengio, "Zero-data learning of new tasks," in *Proc. AAAI*, vol. 1, no. 2, 2008, p. 3.
[19] S. Çetin. (2018). *Zero-Shot Learning*. Accessed: Mar. 1, 2023. [Online]. Available: https://web.archive.org/web/20230301150047/ and https://cetinsamet.medium.com/zero-shot-learning-53080995d45f
[20] Y. Xian, B. Schiele, and Z. Akata, "Zero-shot learning—The good, the bad and the ugly," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4582–4591.
[21] E. Tiu. (2021). *Understanding Zero-Shot Learning—Making ML More Human*. Accessed: Mar. 1, 2023. [Online]. Available: https://web.archive.org/web/20230301151102/https://towardsdatascience.com/understanding-zero-shot-learning-making-ml-more-human-4653ac35ccab?gi=a027f7f662f5
[22] M. K. Yucel, R. G. Cinbis, and P. Duygulu, "How robust are discriminatively trained zero-shot learning models?" *Image Vis. Comput.*, vol. 119, Mar. 2022, Art. no. 104392.
[23] Z. Zhang, Q. Liu, S. Qiu, S. Zhou, and C. Zhang, "Unknown attack detection based on zero-shot learning," *IEEE Access*, vol. 8, pp. 193981–193991, 2020.
[24] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Label-embedding for attribute-based classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 819–826.
[25] E. Kodirov, T. Xiang, and S. Gong, "Semantic autoencoder for zero-shot learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3174–3183.
[26] J. L. Ba, K. Swersky, S. Fidler, and R. Salakhutdinov, "Predicting deep zero-shot convolutional neural networks using textual descriptions," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4247–4255.

[27] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2014, pp. 806–813.
[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.
[29] B. Hu, Z. Lu, H. Li, and Q. Chen, "Convolutional neural network architectures for matching natural language sentences," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 1–9.
[30] A. Tekerek, "A novel architecture for web-based attack detection using convolutional neural network," *Comput. Secur.*, vol. 100, Jan. 2021, Art. no. 102096.
[31] H. Liu, B. Lang, M. Liu, and H. Yan, "CNN and RNN based payload classification methods for attack detection," *Knowl.-Based Syst.*, vol. 163, pp. 332–341, Jan. 2019.
[32] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. ICML*, 2010, pp. 1–8.
[33] M. Komisarek, R. Kozik, M. Pawlicki, and M. Choraś, "Towards zero-shot flow-based cyber-security anomaly detection framework," *Appl. Sci.*, vol. 12, no. 19, p. 9636, Sep. 2022.
[34] W. Rong, B. Zhang, and X. Lv, "Malicious web request detection using character-level CNN," in *Proc. Int. Conf. Mach. Learn. Cyber Secur.* Cham, Switzerland: Springer, 2019, pp. 6–16.
[35] I. Jemal, M. A. Haddar, O. Cheikhrouhou, and A. Mahfoudhi, "Malicious Http request detection using code-level convolutional neural network," in *Proc. Int. Conf. Risks Secur. Internet Syst.* Cham, Switzerland: Springer, 2020, pp. 317–324.
[36] Scikit-learn Developers (BSD License). (2022). *Compare the Effect of Different Scalers on Data With Outliers*. Accessed: Mar. 1, 2023. [Online]. Available: https://web.archive.org/web/20230301151653/https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html
[37] D. M. W. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," 2020, *arXiv:2010.16061*.

DILEK YILMAZER DEMIREL received the B.S. and M.S. degrees from the Department of Computer Engineering, Ege University. She is currently pursuing the Ph.D. degree with Istanbul Technical University. She works professionally on corporate web and mobile application software development in the industry.

MEHMET TAHIR SANDIKKAYA (Senior Member, IEEE) received the B.S. degree in electrical engineering from Istanbul Technical University (ITU), in 2002, and the M.S. and Ph.D. degrees in computer science from the Informatics Institute, ITU, in 2005 and 2015, respectively. He contributed to research efforts on several projects with Katholieke Universiteit Leuven (KU Leuven) Technologiecampus Gent, Belgium, from 2008 to 2010. From 2002 to 2016, he was a Research Assistant with Istanbul Technical University. He was an Invited Research Fellow with the Drakkar Research Group, Grenoble Informatics Laboratory (LIG), Grenoble Institute of Technology (Grenoble INP), University of Grenoble-Alpes (UGA), from 2018 to 2019. Currently, he is an Assistant Professor with the Computer Engineering Department, Istanbul Technical University. His current research interests include computer security, privacy, electronic voting systems, computer networks, electronic health systems, cloud computing security, security protocols, and the IoT security.