MASD: Malicious Web Session Detection Using ML-Based Classifier

Dilek Yılmazer Demirel[a] and Mehmet Tahir Sandıkkaya[b]
Department of Computer Engineering, Istanbul Technical University, Istanbul, Turkey

Keywords: Malicious Web Session Detection, Machine Learning, Classification.

Abstract: The development of web applications and services has resulted in an increase in security concerns, especially in identifying malicious web session attacks. Malicious web sessions pose a significant risk to users, potentially resulting in data breaches, illegal access, and other malicious activities. This study presents an innovative technique for detecting malicious web sessions using a machine learning-driven classifier. To examine the features of web sessions, the suggested technique combines an embedding layer and machine learning approaches. Three different datasets were used in the empirical studies to confirm the effectiveness of the approach. They include a unique compilation of Internet banking web request logs, provided by Yap Kredi Teknoloji, as well as the well-known HTTP dataset CSIC 2010 and the publicly accessible WAF dataset. The experimental results are compared to known approaches such as Random Forest, Convolutional Neural Networks (CNN), Support Vector Machines (SVM), Naïve Bayes, Decision Trees, DBSCAN, and Self-Organizing Maps (SOM). The actual findings demonstrate the superiority of the suggested technique, especially when Random Forest is used as the chosen classifier. The attained accuracy rate of 99.17% surpasses the comparison methodologies, highlighting the approach's ability to efficiently identify and block malicious web sessions.

1 INTRODUCTION

The digital interaction, communication, and commerce landscape has been profoundly reshaped by the widespread integration of web-based applications. As users increasingly navigate these virtual environments, their activities have become pivotal to modern life. However, this rapid technological evolution has not only heralded unprecedented convenience but also ushered in an era of heightened vulnerability. In tandem with this progress, malicious actors have ingeniously exploited the expanding digital infrastructure, thereby imperiling the security and privacy of web users. In particular, attacks intended for web sessions have emerged as persistent threats, thereby necessitating the development of potent and streamlined strategies for their identification and prevention (Mansfield-Devine, 2022; Toprak and Yavuz, 2022). Cybersecurity metrics underscore the gravity of the situation, revealing that a substantial 68% of web applications are susceptible to breaches involving sensitive data (PurpleSec, 2022). Equally concerning is the report indicating that web applications serve as the conduit for 56% of all cyberattacks, 95% of which are driven by financial motives (Mansfield-Devine, 2022). In the face of these mounting challenges, conventional rule-based attack prevention systems are found to be incapable of adapting to swiftly evolving attack vectors. As a result, increasing number of researchers have turned their attention to the realm of Machine Learning (ML)-based solutions as a promising avenue for the detection of malicious HTTP sessions, in pursuit of a more agile and effective defense paradigm.

Clustering algorithms have garnered significant attention among researchers seeking effective solutions for the detection of malicious web sessions. In various studies, clustering algorithms such as DBSCAN, SOM, and Modified ART2 have been employed to distinguish between legitimate and malicious sessions, often coupled with a feature extraction phase (Sun et al., 2020; Stevanovic et al., 2011; Sadeghpour et al., 2021). During the feature extraction phase, these investigations extract a constrained set of features from web sessions, which subsequently serve as input for the clustering algorithms. This method facilitates session-based clustering to discern and isolate malicious sessions. Complementary to these unsupervised methodologies, several supervised approaches have also emerged. These endeavors not

[a] https://orcid.org/0000-0002-4008-4478
[b] https://orcid.org/0000-0002-9756-603X

Yılmazer Demirel, D. and Sandıkkaya, M.
MASD: Malicious Web Session Detection Using ML-Based Classifier.
DOI: 10.5220/0012174800003595
In *Proceedings of the 15th International Joint Conference on Computational Intelligence (IJCCI 2023)*, pages 487-495
ISBN: 978-989-758-674-3; ISSN: 2184-3236

only aim to classify web sessions into malicious and non-malicious categories but also employ feature extraction techniques akin to their unsupervised counterparts (Goseva-Popstojanova et al., 2012). In the context of this discourse, the present paper introduces an innovative approach to the detection of malicious web sessions utilizing a Machine Learning (ML)-based classifier. In a departure from conventional techniques, this approach eschews the feature extraction phase that typically involves a limited feature set. In this manner, this novel approach achieves heightened accuracy and a reduced false-positive rate, thereby fortifying its efficacy in the domain of financial security. Major contributions are summarized in the list below:

- Introducing and implementing a new approach to detect malicious web sessions using the Random Forest as a classifier with the embedding layer, as described in Section 4.
- Implementing benchmark models such as SVM, Naïve Bayes, CNN, Decision Tree, DBSCAN, and SOM, as given in Section 6.
- Using proposed models and benchmarks on three different datasets (a unique financial dataset, the CSIC 2010 HTTP dataset (Giménez et al., 2012), and the WAF dataset (Ahmad, 2017)) to compare performance results after the training and the test steps of the model, as shown in Section 7.

The subsequent sections of this paper are structured as follows: Section 2 provides an overview of related research, outlining current methodologies and outcomes in the realm of malicious web session detection. In Section 3, an exposition is provided regarding the datasets employed, accompanied by a detailed elucidation of how the samples within these datasets are categorized. Section 7 offers a presentation of experimental results and comparative analyses, shedding light on the performance of the proposed approach in contrast to existing methods. Finally, the conclusions and implications drawn from the findings of this study are described in Section 7.2.

2 RELATED WORK

Malicious web session detection has attracted the attention of several studies. Both supervised and unsupervised approaches are experimented to detect malicious sessions. In unsupervised approaches, different clustering algorithms together with a feature extraction method are preferred. For instance, *Sun et al.* proposes an unsupervised learning technique called WSAD to detect anomalies utilizing web session data. The authors realize the limits of standard approaches that rely on labeled data and present an alternative. The WSAD technique involves a feature extraction mechanism that converts raw web session data into meaningful representations and a density-based clustering algorithm that groups comparable sessions. In addition, a session-based outlier identification approach is used to identify sessions that differ considerably from existing clusters (Sun et al., 2020). Another study presents a comprehensive approach for identifying abnormalities in web session data, particularly in dynamic environments. The authors discuss the issue of recognizing anomalous behaviors that may occur as a result of the ever-changing nature of web applications. The suggested method incorporates both web usage mining and clustering techniques to efficiently capture abnormal patterns. It adopts a feature extraction approach that includes multiple session-based and sequence-based elements to describe web sessions thoroughly (Gutiérrez et al., 2010). *Stenovic et al.* suggests a method for unsupervised clustering of web sessions to distinguish between malicious and non-malicious website sessions. This paper addresses the challenge of identifying potential threats and abnormal behaviors without relying on labeled data or prior knowledge of attack patterns. The proposed approach combines feature engineering techniques with a clustering algorithm to group similar web sessions together. The method aims to differentiate between legitimate user behavior and malicious activities by capturing the inherent patterns and similarities within the session data (Stevanovic et al., 2011). Additionally, *Sadeghpour et al.* present another unsupervised approach using the SOM algorithm as a clustering method with an automated feature selection method utilizing gradient boosting (Sadeghpour et al., 2021).

On the other hand, several researchers focus on supervised techniques over web session data. For instance, *Goseva et al.* struggle to identify and distinguish between legitimate and malicious web sessions. The suggested method utilizes machine learning techniques and feature engineering to extract meaningful information from web session data. The researchers employ a variety of features such as session duration, request types, and frequency to represent each session. These features are then used to train a classification model capable of accurately categorizing web sessions as malicious or non-malicious (Goseva-Popstojanova et al., 2012). In addition to this work, other approaches exist to determine whether a web session was created by a legitimate user. *Suchacka et al.* focus on separating sessions by creators, ei-

Table 1: Comparative Analysis: Proposed Approach vs. Related Work.

Research	**Date**	**SL**	**UL**	**Approach**	**Dataset**
Sun et al. (Sun et al., 2020)	2020		✓	Clustering (DBSCAN)	Two Different Web Sites Data
Gutiérrez et al. (Gutiérrez et al., 2010)	2010	✓	✓	Web Usage Mining, Clustering, and Graph Mining	Synthetic Data
Stevanovic et al. (Stevanovic et al., 2011)	2011		✓	Clustering (SOM and Modified ART2)	Dataset obtained using Web-Crawlers
Goseva et al. (Goseva-Popstojanova et al., 2012)	2012	✓		Decision Tree (J48 and PART)	WebDBAdmin and Web 2.0 Datasets collected using these honeypots
Sadeghpour et al. (Sadeghpour et al., 2021)	2021		✓	Clustering (SOM)	Dataset provided by the York University's EECS
Suchacka et al. (Suchacka and Iwański, 2020)	2020		✓	Clustering (aIB Algorithm)	Dataset obtained by real e-commerce website
Proposed Approach	2023	✓		Random Forest	Dataset obtained by real banking website, CSIC 2010, and WAF Dataset

ther users or bots, utilizing an agglomerative Information Bottleneck (aIB) algorithm that aims to extract the most informative features from the traffic data while minimizing redundancy (Suchacka and Iwański, 2020).

The comparison of the suggested approach and other studies is depicted in Table 1. This paper introduces a series of distinctive advancements and contributions that set it apart from existing studies on malicious session detection.

- To begin with, the paper proposes a novel approach that specifically targets the detection of malicious sessions. This innovative method addresses the intricate task of extracting discriminative information from both malicious and benign sessions through the utilization of web requests, surmounting this challenge through the implementation of an embedding layer.
- Moreover, this study pioneers the application of the proposed approach to analyze web session data sourced from the banking industry, an arena that has been relatively underexplored in the context of malicious session detection. The efficacy of the introduced approach is rigorously assessed using a diverse set of benchmark datasets, effectively showcasing its substantial practical potential.

3 DATASETS

Every interaction directed at a web server leaves a trace in the form of chronological web request logs. These logs play a pivotal role later in identifying irregularities within web-based activities, offering a wealth of insights into these interactions. By diligently monitoring and analyzing these records, organizations can uncover anomalies indicative of illicit or harmful behavior. For instance, the sudden appearance of repeated requests for previously unutilized resources by a user may signal nefarious intentions (Toprak and Yavuz, 2022).

A unique dataset sourced from Yapı Kredi Teknoloji (YKT) serves as the foundational dataset for evaluating the proposed model, herein referred to as the "Banking Dataset." This collection encompasses web request logs capturing incoming network traffic to an online banking platform. Comprising over 2 million benign requests and more than 10,000 malicious requests, this dataset employs URI and SessionID attributes for malicious web session identification. The CSIC 2010 HTTP dataset, assembled by the "Information Security Institute" of the Spanish National Council for Research (CSIC), constitutes the second evaluation dataset. With more than 25,000 malicious requests and 36,000 legitimate requests, this dataset is designed to facilitate assessments of web attack defense systems (Giménez et al., 2012). The third dataset, known as the WAF dataset, is an open-source compilation sourced from 30 different Web Application Firewalls (WAFs). It encompasses over 1 million benign requests and more than 40,000 malicious requests, aimed at fostering the development of machine learning models capable of detecting and categorizing WAF attacks (Ahmad, 2017).

In these datasets, individual requests aggregate to form user sessions. In the banking dataset, user ses-

Table 2: Dataset Instances (B: benign M: malicious).

Path	Dataset	B	M
/ngi/index.do	Banking	✓	
/core/.env	Banking		✓
/tienda1/index.jsp	CSIC2010	✓	
/iisstart.htm	CSIC2010		✓
/blueberry	WAF	✓	
/ows-bin/windmail.exe	WAF		✓

sions are constructed using the URI and SessionID fields, while other datasets employ a random selection process for user session formation. To effectively measure the performance of the model under evaluation, a deliberate imbalance is created by following a 20:100 ratio between malicious and non-malicious sessions, as observed in previous studies (Sadeghpour et al., 2021). This choice is influenced by the inherent rarity of malicious sessions in the data, constituting approximately 1% of the total sessions (Sun et al., 2020). Consequently, when the proposed model correctly classifies benign data, it attains an accuracy rate of 99%. However, this seemingly impressive accuracy can be misleading when evaluating both benchmark models and the proposed model, as it doesn't adequately address the challenge of accurately detecting the rare and critical malicious sessions within the dataset. Detailed instances of these session proportions are presented in Table 2.

4 USE CASE SCENERIOS

Many users prefer to use banking web applications for their financial transactions. These transactions consist of money transfers, loan or credit card applications, and various payments. A user's session in web applications begins by performing a request, such as a login. Then, this session is ended by user's leave. If no malicious activity occurs during a user session, this session is labeled as benign. Otherwise, it is a malicious session. Due to the large and dynamic web environment, low-volume attacks, and ever-developing attack techniques, detection via rule-based systems is inappropriate. Resolving this contradiction could be possible with ML-based malicious session detection systems if the accuracy and true positive rate could be increased.

Use Case Scenario 1: Initially, an arbitrary user logs into the website of the bank. Then, they control the balances of their accounts. They want to pay off their credit card debt. They open the credit cards page and click the button for payments. They fill out the amount. Inadvertently, a key on the numeric pad is stuck and a large amount is typed. They click the button for payment operation and create a request to the system. The rule-based prevention system of the bank interrupts the operation and forces a log-out. However, all activities of this session are legal yet a mistake exists. In this case, MASD should provide the correct prediction. Otherwise, interruption of payments will lead to the loss of trust and reputation.

Use Case Scenario 2: An attacker logs into the banking website akin to an arbitrary user. He checks balances and credit cards. Then, he wants to transfer money from this account to another account. He opens the transfer page and fills out all details on this page. However, he traces the request for transfer operation and manually changes destination and source accounts. The updated request is sent to the web application several times. In this case, all requests are passed by a rule-based prevention system as they do not contain large amounts or illegal characters. However, MASD should be able to catch such a session as malicious. Otherwise, financial loss occurs.

5 THE MASD ARCHITECTURE

The focus of this paper lies in introducing a novel approach designed to identify malicious web sessions. The suggested approach (MASD: Malicious Web Session Detection) is a supervised approach that uses the Random Forest algorithm as a classifier with an embedding layer. This approach aims to obtain high accuracy and true positive rate despite the expansive and dynamic nature of the web, low-volume attacks, and data imbalances. Web application attacks are largely preventable with this approach.

The MASD architecture, shown in Fig 1, contains a classifier and embedding layer. The Random Forest algorithm is utilized as a classifier in this approach. The hyperparameters are defined as $n_{estimators} = 10$ and $random_{state} = 2$. Before the data is given to the embedding layer, it is preprocessed using code embedding ($Vector_x = CodeEmbedding(X_{input})$) and is grouped by sessions ($Vector_s = GroupBySession(Vector_x)$). The session list is an input for the embedding layer ($X_{output} = EmbeddingLayer(Vector_s)$). An embedding layer plays a vital role in classification tasks by transforming categorical or discrete data into dense vector representations. It captures semantic relationships, reduces dimensionality, encodes latent features, enhances generalization and robustness, and enables transfer learning. As a result of applying this layer, high accuracy and true positive rate in experimental trials are achieved. The output of the embedding layer is given to a classifier. This algorithm of this approach

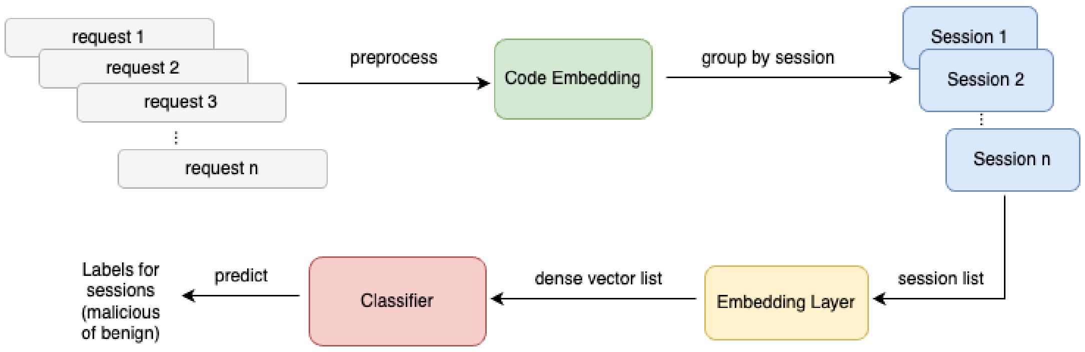

Figure 1: The MASD Architecture: The web requests are given as input for the preprocessing step. All requests are converted to a vector of ASCII codepoints. Next, this data is grouped by sessions. The session list is given to an embedding layer. This layer's output, a dense vector list, is given to the classifier to predict labels for sessions.

is presented in Algorithm 1. It is used as a classifier to predict sessions as malicious or benign. The Random Forest algorithm is an ensemble learning method that combines multiple decision trees to make accurate predictions. Each decision tree in the forest is constructed independently, and the final prediction is obtained through a voting or averaging process. At the end of the classification, the predictions are evaluated using evaluation metrics such as accuracy, recall, precision, and F1 score.

Algorithm 1: MASD: Malicious Web Session Detection.

```
Input: $Dataset$: Labeled dataset
  $Dataset = ((x_1,y_1),...,(x_{m+n},y_{m+n}))$ with $m+n$
  queries
$X_m$, $Y_m$: Malicious samples $X_m = (x_1,...,x_m)$ and
  labels $Y_m = (y_1,...,y_m)$ with m queries
$X_n$, $Y_n$: Benign samples $X_n = (x_1,...,x_n)$ and labels
  $Y_n = (y_1,...,y_n)$ with n queries
Output: $Y_p$: Labels predicted by the classifier
Begin:
Initialize: $X_m \leftarrow [], Y_m \leftarrow [], X_n \leftarrow [], Y_n \leftarrow []$

$BenignQueries, MaliciousQueries \leftarrow$ seperate
  Dataset
$X_n, Y_n \leftarrow$ apply code embedding for
  $BenignQueries$
$X_m, Y_m \leftarrow$ apply code embedding for
  $MaliciousQueries$

$X_{ms}, X_{ns} \leftarrow$ group $X_n$ and $X_m$ by session
$Y_{ms}, Y_{ns} \leftarrow$ set labels according to $X_{ms}, X_{ns}$
$X_{train}, X_{test} \leftarrow$ get samples using $X_{ms}, X_{ns}$
$X_{train}, X_{test} \leftarrow$ process $X_{train}, X_{test}$ using
  embedding layer
$Y_{train}, Y_{test} \leftarrow$ get samples using $Y_{ms}, Y_{ns}$
train classifier with $X_{train}, Y_{train}$
$Y_p \leftarrow$ predict labels using classifier with $X_{test}$

End of algorithm
```

5.1 Data Preprocessing

MASD approach utilizes code embedding as data preprocessing for web request logs (Jemal et al., 2020). The Uniform Resource Identifier (URI) is the first input. It consists of a string to identify a resource on the Internet. Removing the hostname of this string yields the path, the query and the fragment of the URL (used as the path in the rest of the paper for brevity). The code embedding provides that the characters of the path are presented as vectors consisting of ASCII codes. Owed to it, characters are grouped together or compared according to how similar they are. This method is preferred in machine translation and language modeling as a useful approach (Jemal et al., 2020). Table 3 depicts the conversion to ASCII code values from paths. After obtaining the ASCII codes list for each path, the requests are grouped by session. Session ID values are utilized in the Banking Dataset to prepare sessions. Each session includes a maximum of one hundred requests. In other datasets, sessions are prepared by selecting up to a hundred requests.

5.2 Evaluation Metrics

This paper uses the accuracy, recall, precision, and F1 score as evaluation metrics. The prediction is identified as True Positive (TP) if a malicious session (Actual Positive) is labeled as a malicious session (Predicted Positive). True Negative (TN), False Positive (FP), and False Negative (FN) represent other respective results in the same manner. The calculations of evaluation metrics are shown with the following formulas.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{1}$$

$$Recall(TruePositiveRate) = \frac{TP}{TP+FN} \tag{2}$$

Table 3: Embedding of Instances.

Path	ASCII Codepoint Vector Representation
/ngi/index.do	[47, 110, 103, 105, 47, 105, 110, 100, 101, 120, 46, 100, 111]
/ows-bin/windmail.exe	[47, 111, 119, 115, 45, 98, 105, 110, 47, 119, 105, 110, 100, 109, 97, 105, 108, 46, 101, 120, 101]

$$Precision = \frac{TP}{TP+FP} \tag{3}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{4}$$

6 BENCHMARK MODELS

SVM, Naïve Bayes, CNN, Decision Tree, DBSCAN, and SOM are implemented as the benchmark models to evaluate the proposed approach. The experimental trials are carried out on the same datasets.

Support Vector Machines (SVMs) are popular and powerful class of supervised learning algorithms used for classification and regression tasks. SVMs aim to find the optimal hyperplane that separates different classes or predicts continuous values with the maximum margin. They are particularly effective in handling complex and high-dimensional data by transforming it into a higher-dimensional feature space using kernel functions. The key strength of SVMs lies in their ability to handle non-linear relationships by implicitly mapping the data into a higher-dimensional space. By maximizing the margin between classes, SVMs promote generalization and robustness against overfitting. They can handle both binary and multi-class classification problems and provide continuous predictions for regression tasks. However, SVMs can be computationally expensive, require careful selection of hyperparameters, and their performance may vary depending on the choice of the kernel function. Nonetheless, SVMs remain widely used and valued for their flexibility, robustness, and theoretical foundation in machine learning (Azab et al., 2022; Cortes and Vapnik, 1995). The hyperparameters are defined as $\gamma = 10^3$, $C = 0.25$, and $tol = 10^{-7}$.

Naïve Bayes (NB) is a notably uncomplicated yet formidable probabilistic machine learning algorithm employed extensively for classification endeavors. Rooted in Bayes' theorem, this algorithm operates under the presumption that features are conditionally independent given the class variable, a concept termed naïve independence. This assumption simplifies computation, facilitating efficient training and prediction processes. Notable for their rapid learning pace and scalability, Naïve Bayes models prove advantageous for vast datasets. Their robust performance, even when confronted with limited training data and high-dimensional feature spaces, has established their significance. Notably, Naïve Bayes classifiers find widespread application in text classification tasks like spam filtering or sentiment analysis, where the independence assumption aligns well. Although the naivety hypothesis might not hold true in all scenarios, Naïve Bayes classifiers consistently deliver competitive outcomes, embodying a dependable and comprehensible baseline model for classification challenges (Azab et al., 2022; Lewis, 1998).

Convolutional Neural Networks (CNNs) are a subset of deep learning models built to be exceptionally good at processing and evaluating visual data such as images and videos. CNNs have transformed computer vision tasks and attained cutting-edge performance in several fields. The key component of CNNs is the convolutional layer, which applies a set of learnable filters to the input data, capturing local patterns and spatial dependencies. This allows CNNs to automatically learn hierarchical representations of the input, starting from low-level features and gradually building up to more complex and abstract features. The pooling layers further downsample the features, reducing the spatial dimensionality and extracting the most salient information. The stacked convolutional and pooling layers create a powerful feature extractor that is robust to variations in position, rotation, and scale. CNNs are typically followed by fully connected layers that perform classification or regression tasks (Azab et al., 2022). Two convolution layers, a batch normalization layer, a max pooling layer, and a fully connected layer comprise the CNN model used for experimental trials. The Epochs parameter is determined as 40, and the batch size parameter is used as 20. Additionally, the Adam optimizer is utilized with the following parameters; $learningRate = 0.0005$, $beta_1 = 0.9$, $beta_2 = 0.999$, and $amsgrad = False$.

Decision Tree is a supervised learning technique. It aids in decision-making by breaking down complex decisions into a series of simpler ones. The algorithm divides data into subsets depending on several features, much like the branches of a tree, providing a clear path to solutions. Classification trees are tree models where the target variable can take a discrete range of values. The leaves of these tree structures represent class labels obtained from the branches of the attributes (Goseva-Popstojanova et al., 2012). Default hyperparameters are used ($criterion$ =“gini”, $min_{samplesSplit} = 2$, $min_{samplesLeaf} = 1$, and $min_{weightFractionLeaf} = 0.0$).

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is an unsupervised technique that groups the data points according to their proximity. Researchers in data mining and machine learning frequently use the DBSCAN approach. DBSCAN performs well when all clusters are sufficiently dense and well-separated by low-density regions (Sun et al., 2020). The hyperparameters are used as $\varepsilon = 0.19$, $min_{samples} = 2$.

Self-Organizing Map (SOM) is a popular unsupervised machine learning (ML) technique in the data exploration and visualization area. It creates a low-dimensional representation of high-dimensional data by matching a grid of nodes to the input information over a predefined number of repetitions. Typically, neurons are built into a single, rectangular or hexagonal 2-dimensional grid (called a node or reference vector) (Sadeghpour et al., 2021). The hyperparameters are $num_{rows} = 10$, $num_{cols} = 10$, $max_{mdistance} = 4$, $max_{learningRate} = 0.5$, and $max_{steps} = 750$.

All benchmark models are placed as a classifier in the MASD architecture. Then, the experimental results are obtained for benchmark models. Validation and training accuracy values are followed to avoid overfitting. Overfitting occurs when the training accuracy continues to increase while the validation accuracy plateaus or diminishes. The stability of training and validation accuracy values have been provided by tuning hyperparameters. The descriptions of hyperparameters are given in Table 6.

7 RESULTS

The experimental trials are completed on an ordinary computer using Python 3.8[1] and Tensorflow 2.10[2] to implement all models.

7.1 Evaluation with Datasets

For evaluation in the banking domain, the sample session sets are meticulously arranged in proportions of 20 malicious sessions to 100 benign sessions, ensuring a balanced assessment. For all datasets, each session contains between one and a hundred requests. The proposed approach has achieved better results (0.9583) than the benchmark models over the banking dataset. The SOM algorithm has obtained the second-best accuracy (0.9167). Decision Tree has achieved an accuracy of 0.9083. The accuracy of CNN and SVM is 0.8333. NB has better accuracy (0.8417) than CNN and SVM. Additionally, the DBSCAN algorithm has the lowest accuracy of 0.7000.

The sample session sets are generated by using malicious and benign queries for WAF and CSIC 2010 datasets. After compiling malicious and benign sessions, the sample sets of these sessions are prepared with 20:100 proportions (20 malicious sessions and 100 benign sessions). The proposed approach has outperformed benchmark models over CSIC2010 data. The suggested technique has an accuracy of 0.9667, while CNN and SVM have an accuracy of 0.8333. Additionally, NB and DBSCAN have an accuracy of 0.7833, and 0.7417, respectively. The SOM algorithm has an accuracy of 0.9417. Over the WAF dataset, the suggested technique has the highest accuracy (0.9917). The accuracy values of both SVM, CNN, and NB are 0.8333, 0.8417, and 0.6583, respectively. Additionally, DBSCAN has an accuracy of 0.7500. SOM and Decision Tree are very promising, with accuracy rates of 0.9833 and 0.9750. These results are presented in Table 4.

7.2 Summary

Upon analysis of the results presented in Table 4, it becomes evident that the proposed model exhibits a notable performance superiority over its rival methodologies across all examined datasets. The technique securing the second-highest accuracy, Self-Organizing Maps (SOM), falls short by a margin ranging from 1% to 4% when compared to the proposed model's accuracy. Similarly, the Decision Tree method, ranking third in accuracy, lags behind by approximately 2% to 7% across the datasets. The Support Vector Machine (SVM) maintains a consistent accuracy value across all datasets, yet this value remains consistently 12% to 16% lower than that achieved by the proposed model. The Naive Bayes (NB) approach registers a significantly diminished performance, attaining an accuracy of 11% to 30% lower than that of the proposed model. Likewise, the Convolutional Neural Network (CNN) yields accuracy figures that are consistently 12% to 16% below the proposed model's accuracy. Furthermore, the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) demonstrates an accuracy shortfall of 22% in comparison to the proposed approach. These findings collectively underscore the robustness and efficacy of the proposed model in deliv-

[1]Python Software Foundation. Available online: https://www.python.org/

[2]TensorFlow Library. Available online: https://www.tensorflow.org

ering superior results across diverse datasets.

Table 4: Results of the Experiments (Top two results of each metric are bolded, precedence of MASD is certain).

Method	Metric	Banking	CSIC2010	WAF
MASD with RF (Proposed)	**Accuracy**	**0.9583**	**0.9667**	**0.9917**
	Recall	**0.9583**	**0.9667**	**0.9917**
	Precision	**0.9594**	**0.9662**	**0.9917**
	F1 Score	**0.9587**	**0.9660**	**0.9916**
SVM	**Accuracy**	0.8333	0.8333	0.8333
	Recall	0.8333	0.8333	0.8333
	Precision	0.6944	0.6944	0.6944
	F1 Score	0.7576	0.7576	0.7576
Naïve Bayes	**Accuracy**	0.8417	0.7833	0.6583
	Recall	0.8417	0.7833	0.6583
	Precision	0.9188	0.8933	0.8880
	F1 Score	0.8589	0.8092	0.7008
CNN	**Accuracy**	0.8333	0.8333	0.8417
	Recall	0.8333	0.8383	0.8417
	Precision	0.6944	0.6944	0.8669
	F1 Score	0.7576	0.7576	0.7769
Decision Tree	**Accuracy**	0.9083	0.8917	0.9750
	Recall	0.9083	0.8917	0.9750
	Precision	**0.9409**	0.9343	0.9783
	F1 Score	**0.9156**	0.9012	0.9757
DBSCAN	**Accuracy**	0.7000	0.7417	0.7500
	Recall	0.7000	0.7417	0.7500
	Precision	0.7901	0.8521	0.8241
	F1 Score	0.7317	0.7720	0.7748
SOM	**Accuracy**	**0.9167**	**0.9417**	**0.9833**
	Recall	**0.9167**	**0.9417**	**0.9833**
	Precision	0.9133	**0.9455**	**0.9837**
	F1 Score	0.9105	**0.9365**	**0.9830**

8 CONCLUSIONS

In the realm of web session security, this paper represents a significant stride forward by presenting a machine learning-grounded solution for the detection of malicious web sessions. The study introduces a novel classifier-driven methodology that capitalizes on the potency of machine learning algorithms to discern and classify malicious web sessions. This approach is applied across three distinct datasets, underscoring its versatility and potential effectiveness. Notably, the approach stands out by achieving a remarkable accuracy rate, dispelling the need for the often restrictive feature extraction phase that typically extracts a limited array of features. The recorded results strikingly surpass the 99% mark across all assessed metrics. These outcomes collectively highlight the potential efficacy of the machine learning-based classifier approach in fortifying web session security. Furthermore, the groundwork laid by this research paves the way for future extensions, possibly encompassing the classification of diverse user sessions and a deeper exploration of user behavior patterns.

REFERENCES

Ahmad, F. (2017). WAF malicious queries data sets. [Online]. Available: https://web.archive.org/web/20230301151428/https://github.com/faizann24/Fwaf-Machine-Learning-driven-Web-Application-Firewall/. (Accessed on 01 March 2023).

Azab, A., Khasawneh, M., Alrabaee, S., Choo, K.-K. R., and Sarsour, M. (2022). Network traffic classification: Techniques, datasets, and challenges. *Digital Communications and Networks*.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20:273–297.

Giménez, C. T., Villegas, A. P., and Marañón, G. Á. (2012). Information Security Institute, HTTP DATASET CSIC 2010. [Online]. Available: https://www.tic.itefi.csic.es/dataset/. (Accessed on 18 December 2014).

Goseva-Popstojanova, K., Anastasovski, G., and Pantev, R. (2012). Classification of malicious Web sessions. In *2012 21st International Conference on Computer Communications and Networks (ICCCN)*, pages 1–9. IEEE.

Gutiérrez, M. G.-C., Pongilupi, J. V., and LLinàs, M. M. (2010). Web sessions anomaly detection in dynamic environments. In *ISSE 2009 Securing Electronic Business Processes: Highlights of the Information Security Solutions Europe 2009 Conference*, pages 216–220. Springer.

Jemal, I., Haddar, M. A., Cheikhrouhou, O., and Mahfoudhi, A. (2020). Malicious HTTP request detection using code-level convolutional neural network. In *International Conference on Risks and Security of Internet and Systems*, pages 317–324. Springer.

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In *Machine Learning: ECML-98: 10th European Conference on Machine Learning Chemnitz, Germany, April 21–23, 1998 Proceedings 10*, pages 4–15. Springer.

Mansfield-Devine, S. (2022). Verizon: Data Breach Investigations Report.

PurpleSec (2022). Cyber Security Statistics: The Ultimate List Of Stats Data, & Trends For 2022. [Online]. Available: https://web.archive.org/web/20221205155455/https://purplesec.us/resources/cyber-security-statistics/. (Accessed on 5 December 2022).

Sadeghpour, S., Vlajic, N., Madani, P., and Stevanovic, D. (2021). Unsupervised ML Based Detection of Malicious Web Sessions with Automated Feature Selection: Design and Real-World Validation. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–9. IEEE.

Stevanovic, D., Vlajic, N., and An, A. (2011). Unsupervised clustering of Web sessions to detect malicious and non-malicious website users. *Procedia Computer Science*, 5:123–131.

Suchacka, G. and Iwański, J. (2020). Identifying legitimate Web users and bots with different traffic profiles—an Information Bottleneck approach. *Knowledge-Based Systems*, 197:105875.

Sun, Y., Xie, Y., Wang, W., Zhang, S., Gao, J., and Chen, Y. (2020). WSAD: An Unsupervised Web Session Anomaly Detection Method. In *2020 16th International Conference on Mobility, Sensing and Networking (MSN)*, pages 735–739. IEEE.

Toprak, S. and Yavuz, A. (2022). Web Application Firewall Based on Anomaly Detection using Deep Learning. In *ACTA INFOLOGICA 2022*, pages 142–149.

APPENDIX

Table 5 shows the abbreviations utilized in this paper.

The descriptions mentioned in this paper are shown in Table 6.

Table 5: ABBREVIATIONS.

Abbreviation	**Definition**
B	Benign
CNN	Convolutional Neural Network
FN	False Negative
FP	False Positive
HTTP	Hypertext Transfer Protocol
M	Malicious
ML	Machine Learning
RF	Random Forest
SL	Supervised Learning
SOM	Self-Organizing Map (SOM)
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
TPR	True Positive Rate
UL	Unsupervised Learning
URI	Uniform Resource Identifier
WAF	Web Application Firewall
WSAD	Web Session Anomaly Detection
YKT	Yapı Kredi Technology

Table 6: Descriptions of Hyperparameters.

Method	**Hyperparameter**	**Description**
RF	$n_{estimators}$	The number of trees in the forest
	$random_{state}$	The randomization parameter for the bootstrapping of data points used in creating trees and sampling features to use when searching for the optimum split across all nodes
SVM	γ	The coefficient of the kernel
	C	The parameter for regularization
	tol	Criteria for terminating tolerance
CNN	$learningRate$	The learning rate
	$beta_1$	Exponential decay rate of first-moment estimates
	$beta_2$	Exponential decay rate of second-moment estimates
	$amsgrad$	Whether or not to use the AMSGrad variation
Decision Tree	$criterion$	The function for determining the quality of a split
	$min_{samplesSplit}$	The bare minimum of samples needed to separate an inner node
	$min_{samplesLeaf}$	The bare minimum of samples necessary at a leaf node
	$min_{weightFractionLeaf}$	The weighted least proportion of the total weights is essential to be at a leaf node
DBSCAN	ε	The shortest distance among two instances for one to be deemed in the vicinity of the other
	$min_{samples}$	The number of instances that must exist in the vicinity for a point to be called a core point
SOM	num_{rows}	The number of rows in the grid
	num_{cols}	The number of columns in the grid
	$max_{mdistance}$	Max value calculated using the Manhattan distance between two grid points
	$max_{learningRate}$	Maximum learning rate
	max_{steps}	The step size for training iteration