International Journal of Information Security (2023) 22:1989–2013
https://doi.org/10.1007/s10207-023-00729-4

REGULAR CONTRIBUTION

CADS-ML/DL: efficient cloud-based multi-attack detection system

Saida Farhat[1] · **Manel Abdelkader**[2] · **Amel Meddeb-Makhlouf**[1] · **Faouzi Zarai**[1]

Accepted: 30 June 2023 / Published online: 13 July 2023

Abstract
With the increasing adoption of cloud computing, securing cloud-based systems and applications has become a critical concern for almost every organization. Traditional security approaches such as signature-based and rule-based have limited detection capabilities toward new and sophisticated attacks. To address this issue, there has been an increasing focus on implementing Artificial Intelligence (AI) in cloud security measures. In this research article, we present CADS-ML/DL, an efficient cloud-based multi-attack detection system. We investigate the effectiveness of Machine Learning (ML) and Deep Learning (DL) techniques for detecting cloud attacks. Our approach leverages a realistic dataset consisting of both benign and fourteen common attack network flows that meet real-world criteria on the AWS cloud platform. We evaluate eight Intrusion Detection Systems (IDSs) based on ML and DL algorithms, including Decision Tree (DT), Random Forest (RF), Extreme Gradient Boosting (XGBoost), Gated Recurrent Units (GRU), Long Short-Term Memory (LSTM), Stacked LSTM, and Bidirectional LSTM (Bi-LSTM) models. Experimental results demonstrate that the CADS-ML/DL system, specifically the XGBoost model, outperforms the other models, exhibiting an accuracy of 0.9770 and a false error rate of 0.0230. Furthermore, we validate the effectiveness of our proposed XGBoost model on the AWS benchmark CSE-CICIDS2018 dataset, attaining a remarkable accuracy score of 0.9999 and an exceptionally low false error rate of 0.0001. Our findings suggest that AI-based approaches have the potential to detect cloud attacks effectively and contribute to the development of reliable and efficient IDSs for cloud security.

Keyword Cloud computing · CICFlowMeter · Machine learning (ML) · Deep learning (DL) · Multi-attack detection system · CSE-CICIDS2018

1 Introduction

The widespread adoption of cloud computing has completely transformed the way organizations store and handle their data. Nevertheless, it has also given rise to novel cybersecurity challenges that necessitate attention, such as unauthorized access, data breaches, and other malicious activities [1–6].

✉ Saida Farhat
saida.farhat@enetcom.u-sfax.tn

Manel Abdelkader
manel.abdelkader@gmail.com

Amel Meddeb-Makhlouf
amel.makhlouf@enetcom.usf.tn

Faouzi Zarai
faouzifbz@gmail.com

[1] ENET'COM, NTS'COM Research Unit, University of Sfax, Sfax, Tunisia

[2] Tunis Business School, University of Tunis, Tunis, Tunisia

To tackle these issues, Intrusion Detection Systems (IDSs) have been widely adopted as effective solutions. By detecting anomalies in network traffic and recognizing known attack patterns through predefined signatures [7–12], IDSs enable system administrators to respond proactively to potential threats and safeguard the security of their networks.

However, the distributed and complex nature of cloud environments, as well as the involvement of multiple third-party services with varying security protocols, make it challenging for traditional IDSs to effectively monitor network traffic and detect potential threats [13–16].

To overcome this challenge, the use of Machine Learning (ML) and Deep Learning (DL) techniques has shown great promise in enabling more effective detection of suspicious activities in the cloud [17–20]. ML and DL algorithms can process huge volumes of data and detect hidden patterns that might be overlooked by conventional IDSs [21, 22]. They can also adapt to new attack patterns and continuously improve their detection capabilities. Additionally, these techniques

can help reduce false predictions and minimize the workload for system administrators.

The main contribution of Karatas et al. [23] is the implementation of six ML-based IDSs for IoT networks using the CSE-CICIDS2018 dataset. The algorithms used are Adaptive Boosting (Adaboost), Decision Tree (DT), Random Forest (RF), K-Nearest Neighbors (KNN), Gradient Boosting (GB), and Linear Discriminant Analysis (LDA). Additionally, they employed the Synthetic Minority Oversampling Technique (SMOTE) to reduce the imbalance ratio in the dataset and improve the system's efficiency for rarely encountered intrusions. However, this study is limited in its scope as it only evaluates traditional ML algorithms and does not explore the potential benefits of utilizing DL techniques, which have demonstrated superior performance in some previous studies.

Zhou et al. [24] created an intelligent IDS that consists of three main stages: Firstly, they employed the CFS-BA heuristic algorithm to reduce dimensionality. Secondly, they utilized an ensemble approach that combined three algorithms (C4.5, RF, and Forest by Penalizing Attributes (Forest PA)). Finally, a voting technique was used to merge probability distributions of base learners for accurate recognition of cyber-attacks. The proposed approach has been evaluated on several datasets, including NSL-KDD, AWID, and CICIDS2017, and has demonstrated superior performance compared to some related works. Specifically, the authors claimed 99% accuracy for NSL-KDD and CICIDS2017 datasets. However, they did not address computation time, which is critical for real-time detection.

In their study, Kim et al. [25] developed a DL-based IDS that utilizes a Convolutional Neural Network (CNN) to specifically identify denial of service (DoS) attacks. The suggested framework was evaluated on the KDDCUP99 and CSE-CICIDS2018 datasets, achieving over 99% accuracy in binary and multiclass classifications on KDDCUP99 and 91.5% on CSE-CICIDS2018. However, the study's main limitation is its narrow focus on DoS attacks.

Rehman et al. [26] conducted a study to evaluate various classification techniques for the detection of DDoS attacks. They examined Naive Bayes (NB), Recurrent Neural Network (RNN), and Gated Recurrent Units (GRU) algorithms, with the results indicating that the GRU algorithm provided the most accurate detection of DDoS attacks, achieving 99.94% accuracy on the CICIDS2019 dataset. The success of the GRU model was attributed to its ability to enhance memory capacity and training performance while addressing challenges such as overfitting, gradient vanishing, and explosion, by maximizing the conditional probability of the target sequence given the source sequence. Nevertheless, it is worth noting that evaluating the model's performance on a single dataset might not be adequate to generalize its efficiency to other datasets or real-world scenarios.

Seth et al. [27] presented a ML-based model that aimed to reduce latency while maintaining the effectiveness of attack detection. They used a hybrid approach involving RF and Principal Component Analysis (PCA) for feature selection, and applied PCA to the essential features to reduce model complexity and improve prediction latency. They trained the model using the Light Gradient Boosting Machine (Light GBM) algorithm and evaluated its performance on the CSE-CICIDS2018 dataset. The results showed an accuracy of 97.73% and an F1-score of 97.57% with 24 selected features. Nevertheless, the model did not include a comprehensive analysis and discussion of its false positive and false negative rates, which could potentially limit its practical value and applicability since an incomplete assessment of its performance may lead to unreliable results.

Fu et al. [28] suggested a model that combines a Bidirectional LSTM (Bi-LSTM) network with an attention mechanism for network traffic anomaly detection. To address data imbalance issues, they used an adaptive synthetic sampling (ADASYN) algorithm and modified the Stacked Autoencoder (SA) model with a more robust dropout structure as a data downscaling method. The model was evaluated on the NSL-KDD dataset, and the experimental results demonstrate better accuracy and F1-score compared to other existing methods, attaining 90.73% and 89.65%, respectively. However, while Bi-LSTM is faster and less complex than some other DL algorithms, it may still require significant computing resources, making it unsuitable for deployment on low-power devices or in resource-constrained environments.

In [29], Sydney proposed an RNN framework for detecting attacks on the UNSW-NB15 and NSL-KDD datasets. The framework used an eXtreme Gradient Boosting (XGBoost)-based algorithm for feature selection, which helped to reduce the feature space of these datasets. The XGBoost-LSTM model demonstrated outstanding performance on the NSL-KDD dataset, achieving a binary classification test accuracy of 88.13%, a validation accuracy of 99.49%, and a training time of 225.46 s. Meanwhile, the XGBoost-Simple-RNN achieved the best performance on the UNSW-NB15 dataset with a test accuracy of 87.07%. For multiclass classification, the proposed framework achieved a test accuracy of 86.93% on the NSL-KDD dataset, while the XGBoost-GRU obtained a test accuracy of 78.40% on the UNSW-NB15 dataset, outperforming existing methods. However, RNN-based models can be computationally expensive, particularly when training on large datasets. This limitation may restrict their applicability to real-world IDS scenarios where models must be trained and evaluated quickly.

Abdelkhalek and Mashaly [30] addressed the class imbalance issue in the NSL-KDD dataset by implementing two data resampling techniques, ADASYN and Tomek-Links, in conjunction with four DL models, Multi-Layer Perceptron (MLP), Deep Neural Network (DNN), CNN, and

CNN-BiLSTM, to improve the detection rate of minority class attacks. Their CNN model achieved an impressive 99.8% accuracy, outperforming modern binary classifiers. Similarly, in the multi-class classification, their MLP model attained a remarkable accuracy of 99.9%, surpassing advanced multi-class classifiers. These results point toward a promising direction for enhancing the identification of minority classes in imbalanced datasets. However, the study's concentration on the NSL-KDD dataset underscores the necessity for further experimentation on real-world datasets with more imbalanced classes to create more robust and dependable NIDS.

The CSE-CICIDS2018 dataset was the subject of a study done by Wang et al. [31], in which several DL models were utilized for binary and multi-class classification, including DNN, CNN, RNN, LSTM, CNN + RNN, and CNN + LSTM. According to the authors, all DL models achieved high accuracy, with multi-class classification accuracy exceeding 98%. Among the models tested, CNN + LSTM demonstrated the most significant improvement in detection performance, although it had a longer inference time than the individual DNN, CNN, RNN, and LSTM models. Despite the promising results of the proposed DL models, it is worth noting that training these models can be computationally expensive and may require powerful hardware to achieve the reported performance. As a result, deploying these models in resource-constrained environments may not be practical.

Table 1 summarizes the existing literature and demonstrates that both ML and DL techniques can be viable options for achieving effective intrusion detection. However, the accuracy and efficiency of these approaches may vary depending on the dataset used for training and testing. Additionally, some DL-based models can be computationally expensive and may not be suitable for deployment in resource-constrained environments. Therefore, further research is necessary to improve the performance of these models and make them more applicable to real-world scenarios. In summary, both ML and DL techniques offer valuable tools for enhancing the security of cloud networks, but their effectiveness depends on various factors that require further investigation.

In this article, we have made several contributions that can be summarized as follows:

- We built a highly available network infrastructure in AWS that includes vulnerable labs such as DVWA, OWASP Mutillidae II, and SQLi Dhakkan.
- We generated a real-time dataset consisting of various attack traces, including BruteForce, DoS, DDoS, Scanning, Directory Traversal, Command Injection, SQL Injection, and XSS, using CICFlowMeter v4.0.
- We conducted a review of recent research articles on ML and DL-based IDS solutions in cloud environments, discussing the strengths and limitations of the proposed approaches.
- Building upon our review and analysis of various ML and DL models, we successfully developed CADS-ML/DL, an efficient cloud-based multi-attack detection system. At the core of our system lies the powerful XGBoost model, playing a crucial role in achieving high detection accuracy and efficiency.
- CADS-ML/DL empowers real-time detection and categorization of the most significant attacks targeting cloud-based systems. This capability enables swift identification and categorization of attacks, thereby enhancing the response and mitigation process.
- We validated the performance of CADS-ML/DL by rigorously testing it against the benchmark dataset CSE-CICIDS2018 on the AWS platform. This validation process provided evidence of the system's capability to effectively detect and mitigate cloud-based attacks.
- To assess the effectiveness of our proposed system, we conducted a comprehensive evaluation using a diverse set of assessment metrics. These metrics included accuracy, precision, recall, F1-score, False Error Rate, Confusion matrix, and Cross-Entropy Loss Function, providing a comprehensive understanding of the system's performance.
- Finally, we compared CADS-ML/DL with other prominent research studies in the field. Our analysis revealed that CADS-ML/DL achieved the highest accuracy with a negligible false error rate, highlighting its superiority in detecting cloud-based attacks compared to existing solutions.

The structure of this paper is as follows: Sect. 2 presents our methodology. In Sect. 3, we conduct a comparison of the outcomes achieved by our approach with those obtained in previous related works. Finally, in Sect. 4, we conclude our study and suggest potential directions for future research.

2 Methodology

Our research approach, as depicted in Fig. 1, follows a five-phase methodology that incorporates CADS-ML/DL, an advanced multi-attack detection system capable of effectively detecting and categorizing various types of attacks in cloud-based environments.

The first phase involves data collection, where we gather the necessary data for analysis. The second phase is data preprocessing, where the collected data are cleaned and processed to eliminate any errors, inconsistencies, and missing values. In the third phase, we perform feature selection to identify the most significant and pertinent attributes from the preprocessed data. Then, in the fourth phase, we use

Table 1 Existing literature

Authors	Year	Algorithms	Contributions	Limitations
Karatas et al. [23]	2020	DT, RF, KNN, Adaboost, GB, LDA	Involved testing various ML models on the CSE-CICIDS2018 dataset, which yielded promising results in terms of accuracy	Did not evaluate DL approaches for intrusion detection in IOT networks
Zhou et al. [24]	2020	C4.5, RF, and Forest PA	Incorporating feature selection and ensemble classification techniques for multi-attack detection, achieving 99% accuracy on both the NSL-KDD and CICIDS2017 datasets	Computation time, a critical factor in real-time detection, has been overlooked
Kim et al. [25]	2020	CNN	The CNN model efficiently detects DoS attacks in the KDDCUP99 and CSE-CICIDS2018 datasets	Focused only on DoS attacks
Rehman et al. [26]	2021	NB, RNN, GRU	Provided accurate detection accuracy of 99.94% of DDoS attacks within the CICIDS2019 dataset with the GRU model	Limited to the detection of DDoS attacks
Seth et al. [27]	2021	Light GBM	Introducing a ML-based model that not only reduces latency but also maintains an effective attack detection system. The proposed approach utilizes a hybrid method that merges RF and PCA for feature selection, reinforced by the Light GBM algorithm, to attain an outstanding accuracy of 97.73% on the CSE-CICIDS2018 dataset	Lack of comprehensive analysis and discussion of false predictions
Fu et al. [28]	2022	Bi-LSTM	Developed a model that combines the Bi-LSTM algorithm with an attention mechanism to identify attacks within the NSL-KDD dataset	The proposed model required significant computing resources
Sydney et al. [29]	2022	XGBoost, LSTM, GRU, Simple RNN	RNN-based framework for intrusion detection on UNSW-NB15 and NSL-KDD datasets	Training RNN-based models on large datasets can be computationally expensive
Abdelkhalek and Mashaly [30]	2023	MLP, DNN, CNN, CNN-BiLSTM	The MLP model achieved 99.9% accuracy in multi-class classification, thereby improving the detection of attacks belonging to the minor categories within the imbalanced NSL-KDD dataset	Need further testing on real-world datasets with more imbalanced classes

Table 1 (continued)

Authors	Year	Algorithms	Contributions	Limitations
Wang et al. [31]	2023	DNN, CNN, RNN, LSTM, CNN + RNN, CNN + LSTM	Several DL frameworks underwent evaluation with the CSE-CICIDS2018 dataset, yielding accuracy rates of over 98% for both binary and multi-class classifications	May not be suitable for deployment in resource-constrained environments

the selected features to train various ML and DL models including DT, RF, XGBoost, GRU, LSTM, Stacked LSTM, and Bi-LSTM. Finally, in the fifth phase, we evaluate the performance of the trained models using various assessment metrics. The subsequent sections will elaborate on each step in detail.

2.1 Data collection

To ensure a comprehensive investigation of our research topic, we adopted a dual approach to data collection. Firstly, we utilized the CSE-CICIDS2018 dataset, which we selected based on its relevance and extensive coverage of our research area. Secondly, we created a custom dataset for web penetration testing purposes on the AWS cloud platform, which is detailed in the following sub-sections.

2.1.1 Experimental environment setup

This section provides a detailed description of our experimental environment, which involved the deployment of a highly available network infrastructure on AWS as shown in Fig. 2.

Our architecture consists of a custom Virtual Private Cloud (VPC) with four subnets: two "public" and two "private," distributed over two availability zones. Each availability zone contains a public and private subnet. We specified two public subnets to increase the availability of the load balancer, which routes traffic to target instances within the private subnets. We launched a single EC2 instance in the private subnet of each availability zone and created a single bastion host (Ubuntu) to establish SSH connections with these instances. To automate the process of updating distribution software packages and installing required packages for a web server, PHP, and MariaDB on instances located in private subnets, we specified User Data instructions. These instructions also included adding the ec2-user to the Apache group, setting the appropriate file ownership and permissions for the web directory and its contents, and creating a basic web page to verify the web server and PHP engine. By specifying these user data instructions, we were able to streamline the setup process for our system, saving time and effort. The security groups for these instances allow SSH, HTTP, and HTTPS access. Once we configured our public and private instances, we created and configured an application load balancer (ALB). After creating the ALB, we verified that it sends traffic over the EC2 instances by checking their status, which should be marked as "healthy" if everything is working correctly. As we have an application with a variable workload, we use the AWS Auto Scaling service. This service automatically sizes resources based on the chosen sizing policy, enabling us to maintain performance and pay only for the resources we really need. It periodically checks the status of its instances and if an instance becomes faulty, it terminates it and replaces it with a new one. In our architecture, the load balancer distributes traffic between instances in the EC2 Auto Scaling Group, enhancing the scalability and availability of our application.

We finish this step by confirming the successful launch of our EC2 instances in the EC2 Dashboard.

2.1.2 Penetration testing labs setup

Before setting up any vulnerable web application penetration testing laboratories on the bastion EC2 instance, we need to install some basic dependencies. The first step is to install Apache, which is the most frequently used web server on Linux systems. Once Apache is installed, we need to edit our instance's firewall inbound rules to allow external users to browse the content on our web server using HTTP or HTTPS.

Apache web servers are commonly paired with the MySQL database engine, as well as scripting languages such as PHP, Python, and Perl, to create a highly capable and resilient environment for building and launching web-based applications.

Once all the necessary dependencies are installed, we can proceed to install the following penetration testing laboratories:

2.1.2.1 Damn vulnerable web application (DVWA) We install the DVWA, which is a free and open-source vulnerable web application developed by Ryan Dewhurst using PHP and MySQL. The purpose of DVWA is to provide a

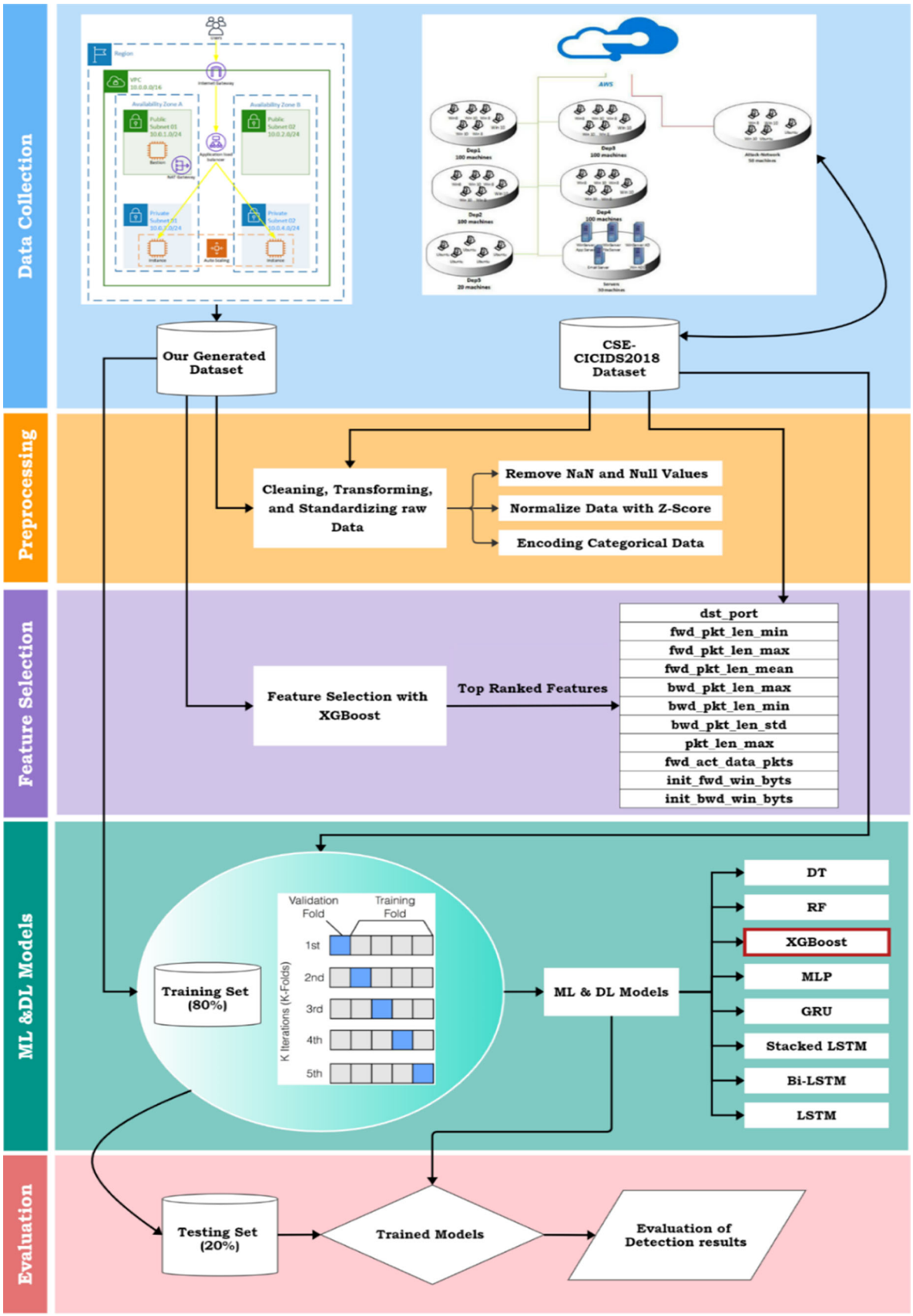


Fig. 1 CADS-ML/DL Methodology Diagram

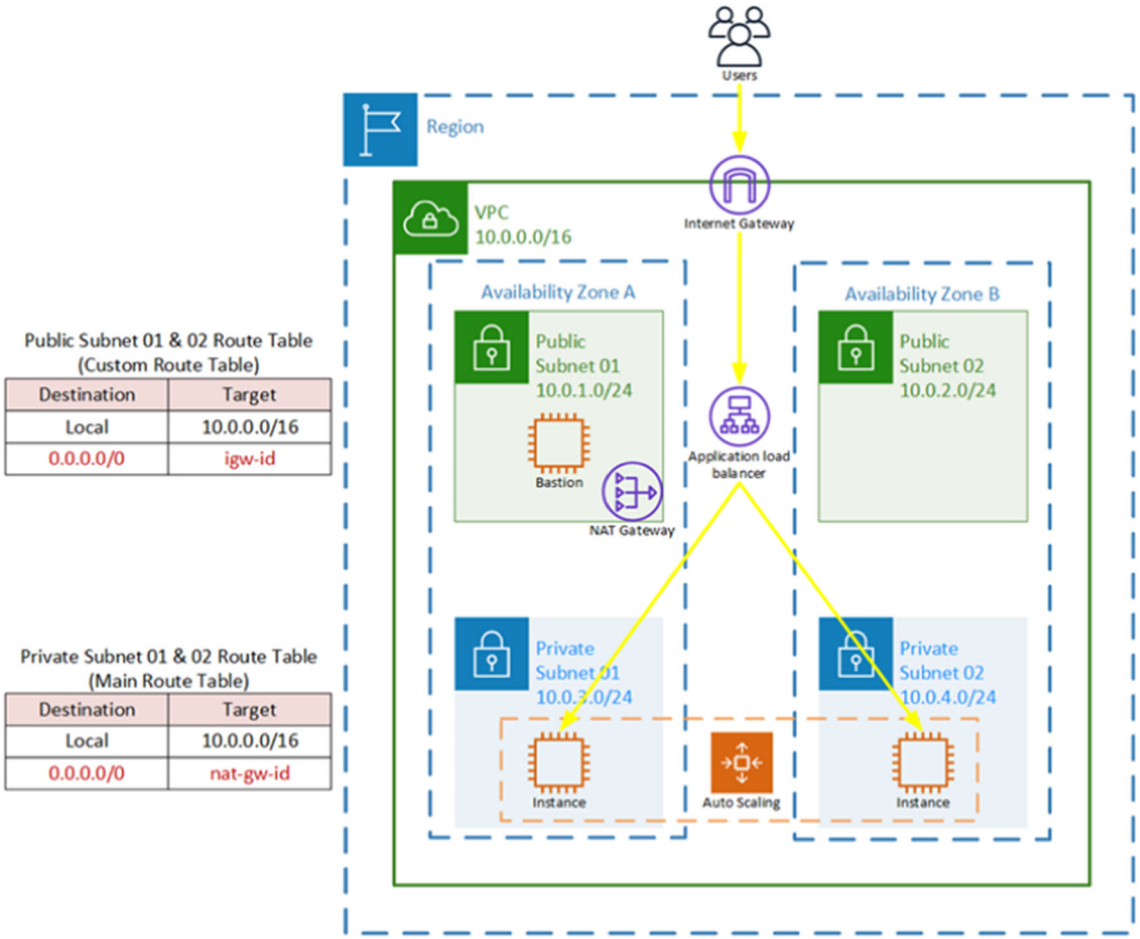


Fig. 2 Highly Available Network Infrastructure on AWS

realistic platform for testing and improving web application security skills. It includes a variety of vulnerabilities and weaknesses that are frequently encountered in real-world web applications, such as SQL injection (SQLi), cross-site scripting (XSS), command injection, file inclusion, and much more.

2.1.2.2 SQL injection – Dhakkan We set up Dhakkan, a deliberately vulnerable web application developed by the Indian security researcher Aung Khan to offer a comprehensive test environment for SQLi attacks. Figure 3 illustrates our access to different kinds of SQLi challenges provided by the Dhakkan Laboratory.

2.1.2.3 OWASP mutillidae II Finally, we proceed with the installation of OWASP Mutillidae II (or simply Mutillidae), which is an open-source and free vulnerable web application developed by OWASP (Open Web Application Security Project) for security testing purposes.

2.1.3 Data generation

To ensure the effectiveness and robustness of our CADS-ML/DL, it was necessary to generate a diverse and extensive dataset consisting of realistic attack scenarios. In this subsection, we detail the attack scenarios that we generated on the AWS cloud platform, which allowed us to supplement the existing CSE-CICIDS2018 dataset for better training and evaluation of our proposed system.

2.1.3.1 Scanning One of the primary steps we take in the exploitation of the vulnerable labs detailed in subSect. 2.1.2 is enumerating hidden directories and files to discover sensitive information about the website. We used the dirb tool developed by Dark Raver to identify critical web content, such as the contents of the http://3.232.110.72/mutillidae/passwords/ directory, as illustrated in Fig. 4.

Another tool we used during the initial stages of our web application security assessment is Nmap, an open-source vulnerability scanner commonly known as "Network Mapper."

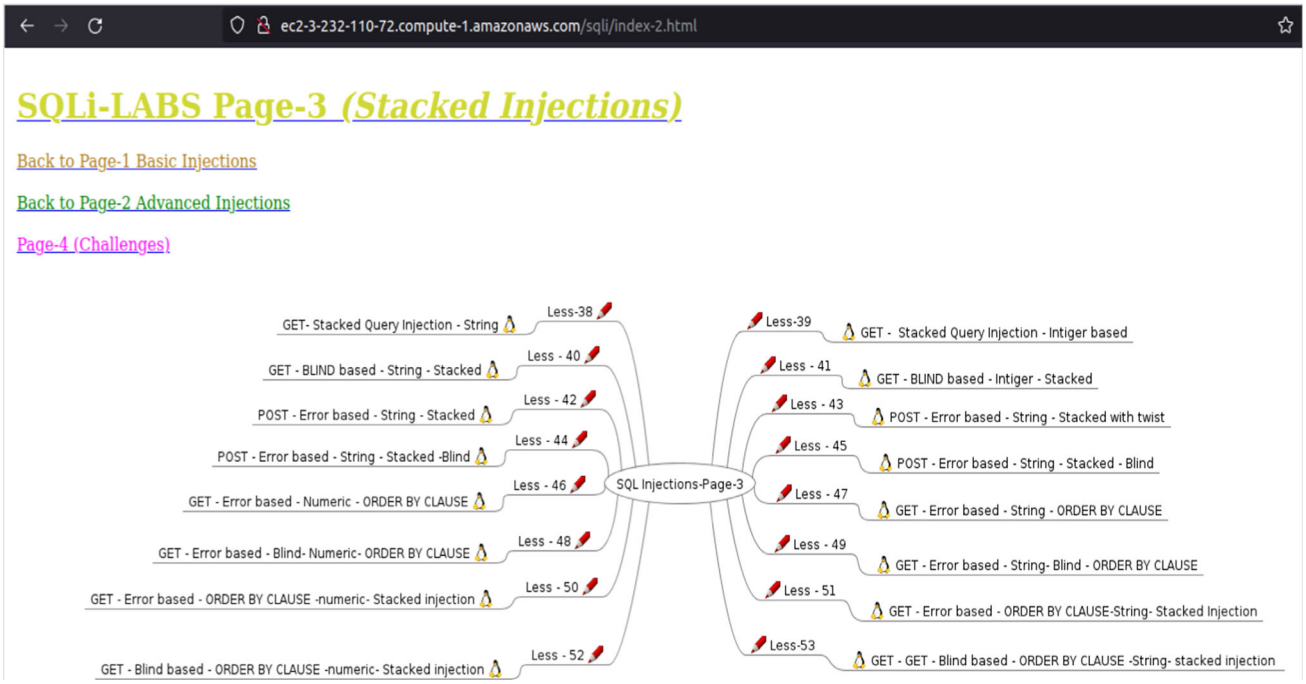


Fig. 3 Dhakkan SQLi-Labs

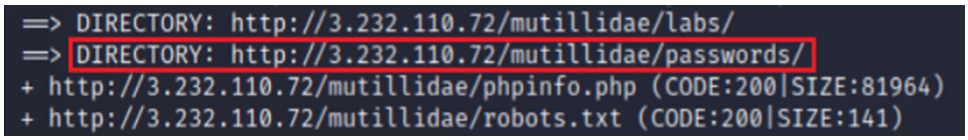


Fig. 4 Hidden passwords directory is identified in DIRB Scan

With the Nmap Scripting Engine (NSE), we were able to identify potential vulnerabilities present in the target system.

2.1.3.2 Directory traversal Directory traversal or file path traversal is a security vulnerability in web applications that enables attackers to access files on the server arbitrarily.

We exploit this vulnerability to reveal the hidden information of the /mutillidae/passwords/ system file. As shown in Fig. 5, we were able to obtain the credentials of the compromised web server.

2.1.3.3 DoS and DDoS In this work, we conducted tests on both DoS and DDoS scenarios. For DoS attacks, we utilized Slowloris, Low Orbit Ion Cannon (LOIC), High Orbit Ion Cannon (HOIC), and TCP SYN Flood as our primary tools. With these types of attacks, a single attacking machine is enough to cause web servers to become completely inaccessible. Slowloris creates a complete TCP connection with the target server and sends valid but incomplete HTTP requests at regular intervals to keep the sockets from closing. As web servers have a limited capacity for serving connections, all of the sockets can be occupied, preventing new connections from being established. HOIC is more advanced than LOIC, designed to work using HTTP floods only. It can attack up to 256 domains simultaneously using a large number of threads, while TCP SYN Flood aims to consume resources on the victim server by sending multiple TCP connection requests faster than the server can process them. Figure 6 shows the number of packets per second sent by LOIC.

For DDoS attacks, we used the HTTP Unbearable Load King (HULK) attack, which exhausts web servers' resources by incessantly requesting one or multiple URLs from numerous attacking machines. The HULK Flood generates a unique pattern with each request to increase the load on the servers and evade intrusion detection and prevention systems. DDoS Hammer is a slow-rate DDoS attack tool that is highly disruptive on most Apache servers. Rather than leveraging large amounts of attack bandwidth or HTTP requests per second, it exploits the maximum current connection time the targeted Apache server can handle. We also tested DDoS Ripper, a disruptive tool that can cut off a server by flooding it with internet traffic.

2.1.3.4 Command injection Command Injection is a type of vulnerability that arises when untrusted user input is passed to a command shell without proper validation. In DVWA, command injection can be found in the "Command Injection" section under the "Vulnerabilities" tab. To exploit this

Fig. 5 Accessing all customer credentials via directory traversal attack

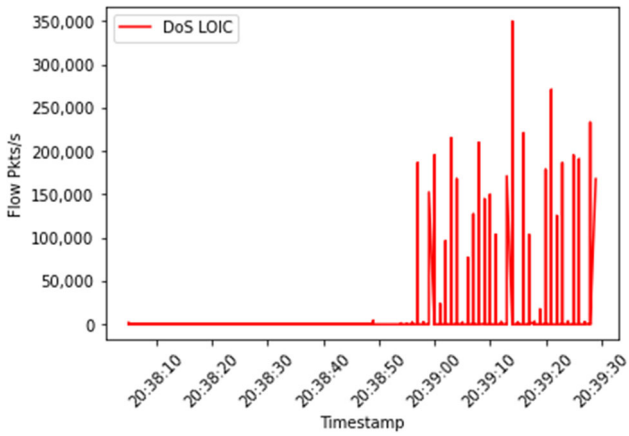


Fig. 6 Flow packet rate of LOIC

vulnerability, we insert a specially crafted input that includes a command injection payload. Since the web application fails to properly validate the input, the injected command is executed on the web server, potentially allowing us to run arbitrary code and gain unauthorized access to sensitive information.

2.1.3.5 SQL injection (SQLi) SQL injection is a type of vulnerability that arises when untrusted user input is not properly sanitized and is passed to a SQL query. In this project, we performed SQLi attacks using both manual and automated approaches. For automated attacks, we utilized SQLMap, a powerful open-source tool capable of supporting multiple database management systems, such as MySQL, Oracle, PostgreSQL, and Microsoft SQL Server. SQLMap can be used to perform a variety of tasks, such as finding SQL injection vulnerabilities in a web application, enumerating the databases and tables in a vulnerable system, dumping the data from a vulnerable database or table, and executing custom SQL statements on a vulnerable database. The results of our SQL injection attacks using SQLMap in the Dhakkan laboratory are illustrated in Fig. 7.

2.1.3.6 Cross-site scripting (XSS) We performed both stored and reflected XSS (Cross-Site Scripting) attacks on the DVWA laboratory. In the stored XSS attack, we injected a malicious script that was later stored in the DVWA application's database and executed every time the exploited page was requested. Figure 8 displays a popup confirming that the stored XSS attack was successful.

In the reflected XSS attack, which is a type of vulnerability that occurs when an application includes unvalidated and unencoded user input in its response, we injected a specially crafted script into the vulnerable DVWA application. The malicious script is then reflected back to the normal user's browser, and if executed, it could potentially allow us to steal the victim's sensitive information, particularly their login credentials.

Database: security
Table: users
[14 entries]

id	password	username
1	Dumb	Dumb
2	I-kill-you	Angelina
3	p@ssword	Dummy
4	crappy	secure
5	stupidity	stupid
6	genious	superman
7	mob!le	batman
8	admin	admin
9	admin1	admin1
10	admin2	admin2
11	admin3	admin3
12	dumbo	dhakkan
13	admin4	admin4
14	admin5	admin5

Fig. 7 SQLi in Dhakkan using SQLMap

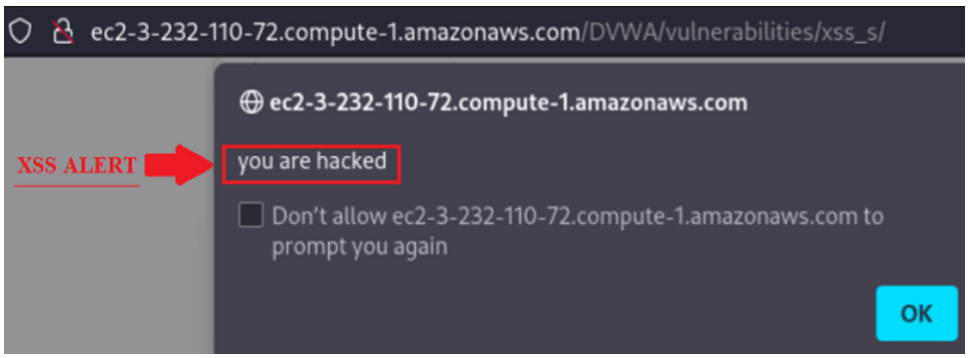
2.1.3.7 BruteForce A BruteForce is a cyber-attack that involves an attacker attempting all possible combinations of characters to guess a password or encryption key, often using automated tools. This attack method can be time-consuming and resource-intensive but can be effective if the password or key is weak or has not been properly secured. Several tools are available that can be used to launch a BruteForce attack. In this work, we used Hydra and Burp Suite, which are web application security testing tools that support a wide range of attacks, including BruteForce attacks. Burp Suite's Intruder module automates the process of sending multiple requests with different parameters, as depicted in Fig. 9. After successfully performing the BruteForce attack, we were able to access the targeted accounts using the username and password combinations discovered by both Hydra and Burp Suite.

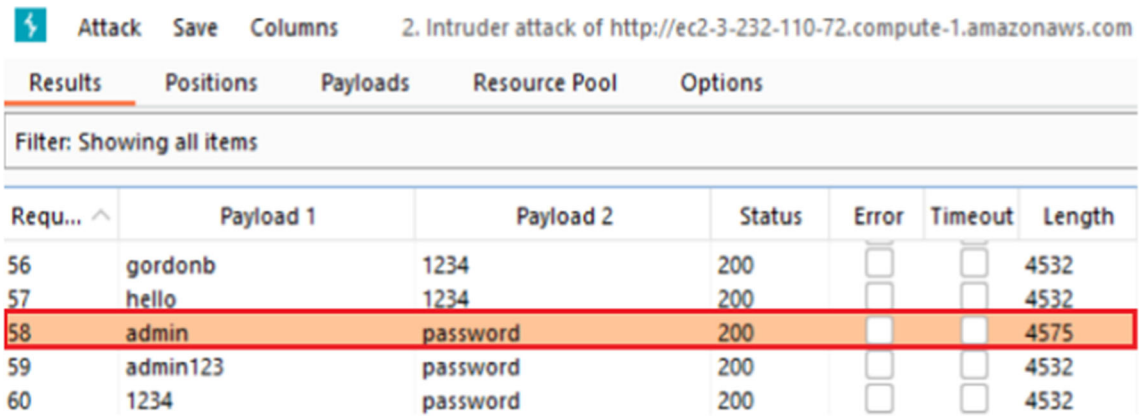
2.1.4 Details of the collected datasets

This subsection offers a comprehensive overview of the datasets employed in our study, namely the generated dataset and the CSE-CICIDS2018 dataset. We conducted a thorough examination of each dataset, including the number of records, classification labels, and feature categories.

2.1.4.1 Details of the generated dataset The dataset used in this study consists of network traffic data generated by a Python version of CICFlowmeter-V4.0, previously known as ISCXFlowMeter. It comprises 87,068 records, each labeled

Fig. 8 Stored XSS Alert





Requ...	Payload 1	Payload 2	Status	Error	Timeout	Length
56	gordonb	1234	200			4532
57	hello	1234	200			4532
58	admin	password	200			4575
59	admin123	password	200			4532
60	1234	password	200			4532

Fig. 9 BruteForce on DVWA with Burpsuite

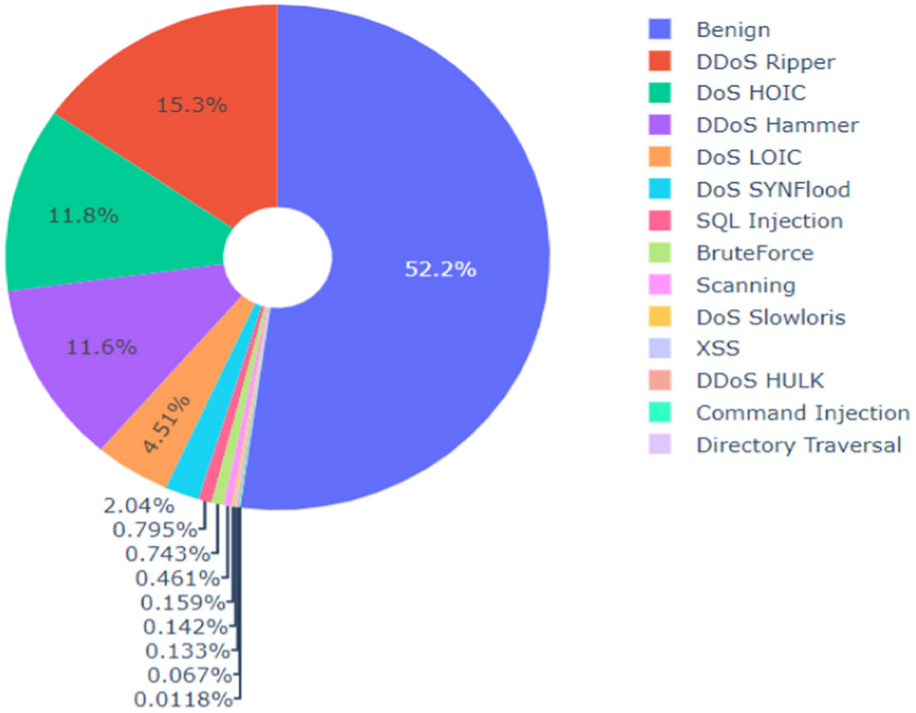


Fig. 10 Distribution of our generated dataset

based on the attack scenario scheduled in conjunction with the IP addresses, ports of both the source and destination, as well as the protocol used. The dataset is distributed across the following labels: Benign (44,394 records), DDoS Ripper (13,043 records), DoS HOIC (10,041 records), DDoS Hammer (9,870 records), DoS LOIC (3,835 records), DoS SYNFlood (1,735 records), SQL Injection (676 records), BruteForce (632 records), Scanning (392 records), DoS Slowloris (135 records), XSS (121 records), DDoS HULK (113 records), Command Injection (57 records), and Directory Traversal (10 records), as illustrated in Fig. 10.

Our dataset consists of 85 features grouped into seven categories based on the type of information they provide as summarized in Table 2.

2.1.4.2 Details of the CSE-CICIDS2018 dataset The second dataset being examined to evaluate the proposed framework of this study is the CSE-CICIDS2018 [32]. It was created in cooperation with Amazon Web Services (AWS), the Canadian government's Communications Security Establishment (CSE), and the Canadian Institute for Cybersecurity (CIC). It is currently the most comprehensive and largest publicly available intrusion detection dataset. The dataset consists of benign and attack traffic that was captured over a period of ten days and is stored in comma separated values (CSV) file format. The traffic was captured using CICFlowMeter-V3, and 80 features were extracted from the captured traffic. The last item of each sample data within the CSE-CICIDS2018 determines whether the packets sent are malicious or not making it suitable for evaluating intrusion detection systems and developing both ML and DL models. There exist six different attack scenarios in this dataset [33] including: Brute-force (380,949 records), Botnet (286,191 records), DoS (654,301 records), DDoS (687,742 records), Web attacks (928 records), and infiltration (161,934 records) of the network from inside. The distribution of class labels used in this paper is depicted in Fig. 11.

2.2 Data preprocessing

Data preprocessing involves cleaning, transforming, and standardizing raw data to remove any errors, inconsistencies, and missing values that may affect the quality, reliability, and validity of the results.

The datasets discussed earlier underwent preprocessing, including the removal of any data samples with NaN and null values and the encoding of categorical labels as integers using the LabelEncoder class from the sklearn.preprocessing module to ensure compatibility with ML and DL algorithms. Additionally, the input values were standardized using z-score normalization according to Eq. 1, as it helps the algorithms converge faster and produce more accurate results compared to min–max scaling.

$$z = (x - \mu) / \sigma \tag{1}$$

where x represents the original value, μ denotes the mean of input data, and σ signifies the standard deviation of the data.

Table 2 Categorization of the features within our generated dataset

Features	Description
src_ip, dst_ip, src_port, dst_port, src_mac, dst_mac, protocol	Basic features of network connections
Timestamp, flow_duration, fwd_iat_tot, bwd_iat_tot	Time-based features
flow_byts_s, flow_pkts_s, fwd_pkts_s, bwd_pkts_s, tot_fwd_pkts, tot_bwd_pkts, totlen_fwd_pkts, totlen_bwd_pkts, fwd_act_data_pkts, init_fwd_win_byts, init_bwd_win_byts, fwd_header_len, bwd_header_len	Features of network bytes/packets
fwd_pkt_len_max, fwd_pkt_len_min, fwd_pkt_len_mean, fwd_pkt_len_std, bwd_pkt_len_max, bwd_pkt_len_min, bwd_pkt_len_mean, bwd_pkt_len_std, pkt_len_max, pkt_len_min, pkt_len_mean, pkt_len_std, pkt_len_var, flow_iat_mean, flow_iat_max, flow_iat_min, flow_iat_std, fwd_seg_size_min, bwd_iat_max, bwd_iat_min, bwd_iat_mean, bwd_iat_std, fwd_iat_max, fwd_iat_min, fwd_iat_mean, fwd_iat_std, pkt_size_avg, active_max, active_min, active_mean, active_std, idle_max, idle_min, idle_mean, idle_std, fwd_byts_b_avg, fwd_pkts_b_avg, bwd_byts_b_avg, bwd_pkts_b_avg, fwd_blk_rate_avg, bwd_blk_rate_avg, fwd_seg_size_avg, bwd_seg_size_avg, down_up_ratio	Statistics of network flows
fwd_psh_flags, bwd_psh_flags, fwd_urg_flags, bwd_urg_flags, fin_flag_cnt, syn_flag_cnt, rst_flag_cnt, psh_flag_cnt, ack_flag_cnt, urg_flag_cnt, ece_flag_cnt, cwe_flag_count	Flag-based traffic features
subflow_fwd_pkts, subflow_bwd_pkts, subflow_fwd_byts, subflow_bwd_byts	Features of network subflows
Label	Content-related traffic features

2.3 Feature selection

In this subsection, we describe our approach to feature selection, where the primary objective was to identify the most relevant features from the given datasets. To streamline our analysis and focus on capturing the essential characteristics of different attack types, we deliberately dropped several columns from the DataFrame, namely "src_ip", "dst_ip", "src_mac", "dst_mac", and "timestamp."

To enhance our feature selection process, we leveraged the XGBoost algorithm, which played a pivotal role in our study. By utilizing a pre-trained XGBoost model and initializing the SelectFromModel method, we effectively transformed the feature matrix. This transformation allowed us to select only the most influential features while discarding less impactful ones. To achieve this, we generated a boolean mask and iterated through a loop to extract the names of the selected features, further refining our focus and ensuring the inclusion of the most crucial attributes.

As a result, we obtained a comprehensive list of the most critical features, which are documented in Table 3.

To visualize the importance of these selected features for our models, we plotted their importance scores in a horizontal bar chart, as shown in Fig. 12. This visualization assists us in deciding which features to include in our models, potentially improving their performance by reducing overfitting and increasing overall efficiency.

By employing this systematic methodology, we ensured that our analysis focused on the most informative and relevant features. This approach led to more reliable findings and valuable insights into the detection and characterization of various attack types. Building upon this strong foundation, the upcoming subsection will explore different ML and DL models. These advanced techniques leverage the power of the carefully selected features, allowing us to delve even further into the intricacies of attack patterns.

2.4 ML and DL models

The ML and DL models in this paper were implemented using Python 3.9.12 programming language in a conda environment with version 4.12.0. Several Python libraries were utilized, including scikit-learn, pandas, NumPy, and Tensorflow, among others. The hardware environment used for this work is an Intel® i7-11370H Central Processing Unit (CPU)

Table 3 Description of the 11 top ranked features

Features	Description
dst_port	Destination port number in the network traffic
fwd_pkt_len_min	Minimum length of packets sent in the forward direction
fwd_pkt_len_max	Maximum length of packets sent in the forward direction
fwd_pkt_len_mean	Mean length of packets sent in the forward direction
bwd_pkt_len_max	Maximum length of packets sent in the backward direction
bwd_pkt_len_min	Minimum length of packets sent in the backward direction
bwd_pkt_len_std	Standard deviation of packet lengths sent in the backward direction
pkt_len_max	Maximum length of all packets sent in the network traffic
fwd_act_data_pkts	Number of active data packets sent in the forward direction
init_fwd_win_byts	Number of bytes sent in initial window of the forward direction
init_bwd_win_byts	Number of bytes sent in initial window of the backward direction

powered laptop, with a processor frequency of 3.30 GHz. The system is equipped with 16 GB of Random-Access Memory (RAM) and NVIDIA GeForce RTX 3050 Ti Graphics Processing Unit (GPU). These specifications make it an excellent choice for our experimentation.

As part of this study, we randomly divided each dataset into a training set consisting of 80% of the original data and a testing set of 20%. To preserve the distribution of the target variables in both sets, we applied the stratified sampling process which involves dividing the dataset into groups or strata based on the values of the target variable. Each stratum represents a distinct category or class of the target variable. By randomly selecting a representative sample from each stratum, the resulting training and testing sets maintain the distribution of the target variable, even in the presence of imbalanced classes. The train and test dataset sizes for our generated dataset were 68,043 and 17,011 rows, respectively, out of a total of 85,054 rows. For the CSE-CICIDS2018 dataset, the train and test dataset sizes were 1,961,641 and 490,411 rows, respectively, out of a total of 2,452,052 rows. Subsequently, we meticulously curated a selection of machine learning (ML) and deep learning (DL) models that align with our research objectives.

2.4.1 ML models

Within the ML models, we deliberately selected Decision Tree (DT), Random Forest (RF), Extreme Gradient Boosting (XGBoost), and Multilayer Perceptron (MLP) algorithms for their distinct strengths in intrusion detection. DT and RF models were chosen for their interpretability, providing insights into the decision-making process. XGBoost leveraged ensemble learning techniques for improved performance, while MLP's nonlinear modeling capabilities enabled us to capture complex relationships within the data. By incorporating these ML models, we aimed to exploit their respective strengths and achieve robust intrusion detection outcomes.

2.4.1.1 Decision tree (DT) DT is a nonparametric classifier that learns from training data. It consists of decision nodes and leaves. Decision nodes split the data based on specific conditions, while leaves provide the final output.

DTs are graphical representations resembling tree structures, serving as flowcharts for decision-making. An illustration of a DT is provided in Fig. 13.

- The root node, or parent node, initiates the tree and splits into decision nodes based on feature values.
- Decision nodes further branch out, allowing for more decision-making.
- Leaf nodes, or terminal nodes, represent the endpoints of the DT. They cannot be split further and provide the final output or class label for a specific path within the tree.
- Sub-trees are branches or subdivisions of the complete DT. They consist of the root node, decision nodes, and leaf nodes, forming smaller and more focused sections of the overall tree.

DTs excel in identifying significant variables and their relationships. They can handle both categorical and numerical data without extensive preparation. However, overfitting and instability are potential drawbacks. Techniques like boosting and bagging can address these issues. Despite these limitations, DTs remain highly valued in intrusion detection due to their interpretability and computational efficiency, making them widely used in various applications within the cloud environment.

2.4.1.2 Random forest (RF) RF is an ensemble learning method that combines multiple DTs to enhance prediction accuracy. Each DT in the RF ensemble is trained on a random subset of the original dataset using bootstrapping, which allows the model to capture different aspects of the data and

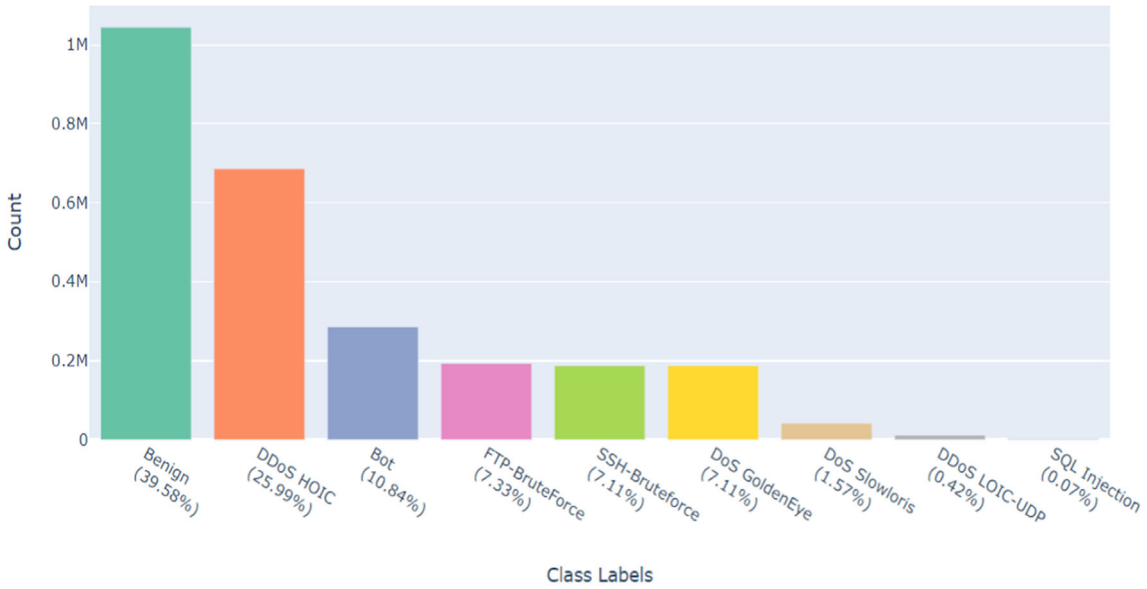


Fig. 11 Distribution of class labels within the CSE-CICIDS2018 dataset

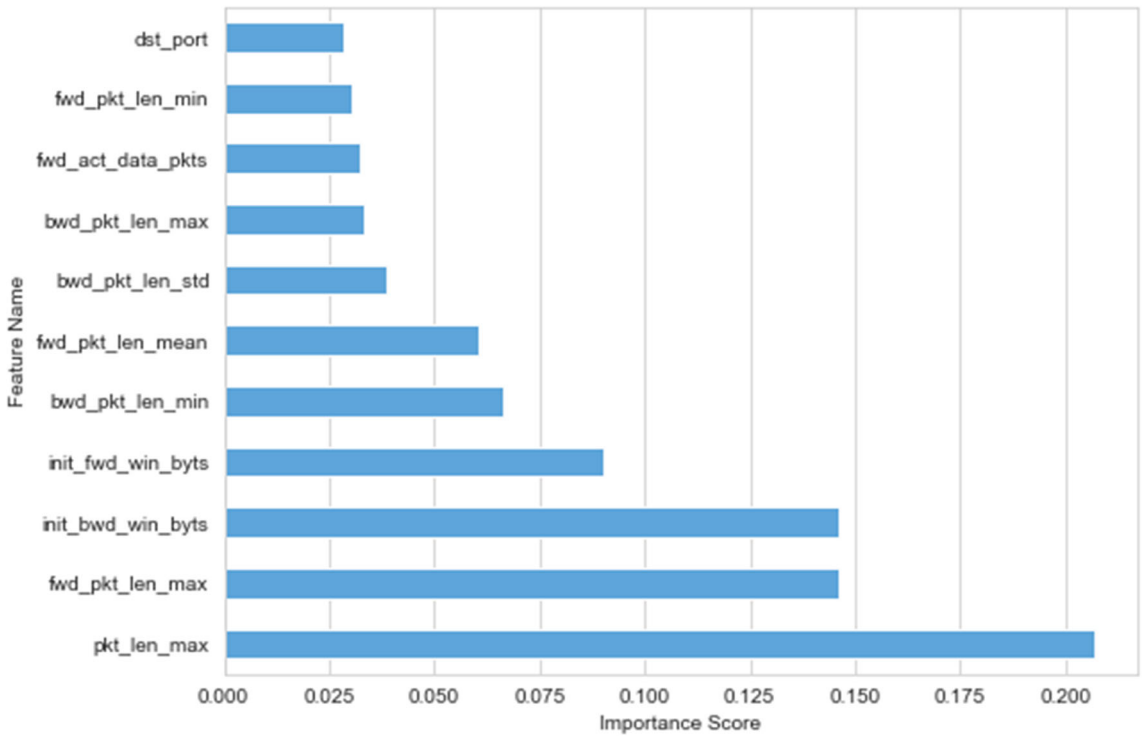


Fig. 12 Top-Ranked Attributes with XGBoost for Models Training and Testing

improve generalization capabilities. Additionally, during the construction of each tree, only a subset of features is considered at each split, reducing correlation between the trees and enhancing RF's robustness and resistance to bias. RF is efficient in handling large datasets and can parallelize the training process. By aggregating the predictions of multiple trees through majority voting, RF significantly improves classification accuracy and provides a reliable means of distinguishing between normal and intrusive network traffic. These advantages, such as improved accuracy, robustness, and scalability, make RF a popular choice for real-time intrusion detection systems.

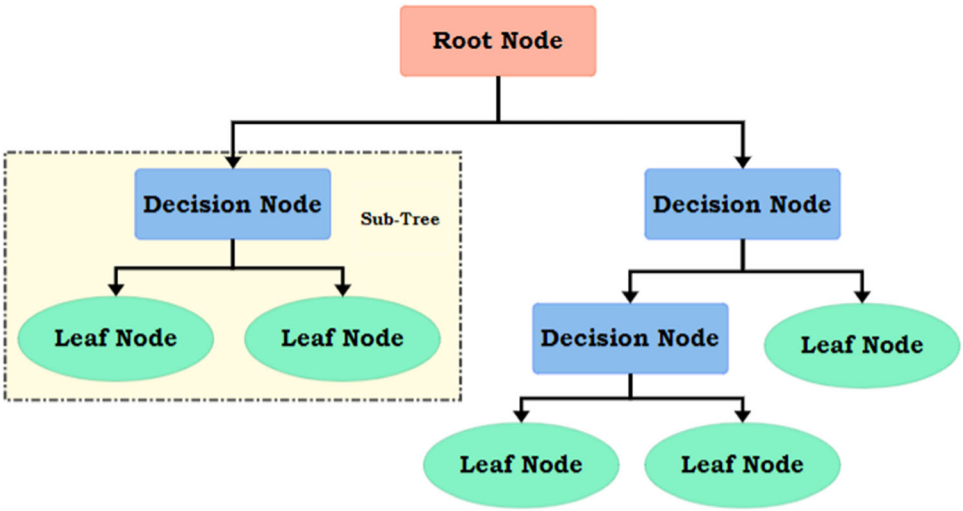


Fig. 13 Illustration of a Decision Tree

2.4.1.3 Extreme gradient boosting (XGBoost) XGBoost is an advanced gradient-boosting framework that combines DTs in an iterative and adaptive manner to enhance predictive accuracy. It is highly scalable, making it suitable for large datasets. XGBoost is computationally efficient, optimizing memory usage and providing faster training and prediction times compared to traditional methods. It supports different data types, including numerical, categorical, and sparse data, automatically handles missing values, and incorporates built-in mechanisms for feature selection and regularization, contributing to improved model generalization and preventing overfitting. Given these capabilities, XGBoost proves to be a highly capable and adaptable tool for real-time multi-attack detection.

2.4.1.4 Multilayer perceptron (MLP) MLP is a widely used neural network model in intrusion detection within the cloud environment. It excels at learning complex patterns and relationships in both categorical and numerical data, making it ideal for analyzing diverse network traffic features. Through its interconnected layers of neurons, MLP captures intricate dependencies and extracts high-level representations from input data. It offers the advantage of flexibility, allowing for the incorporation of various activation functions, optimization algorithms, and network architectures. Additionally, MLP's ability to parallelize computations suits it for cloud environments dealing with large-scale data.

2.4.2 DL models

We specifically employed four DL algorithms, namely Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), Stacked LSTM, and Bidirectional LSTM (Bi-LSTM). These algorithms are all variants of the Recurrent Neural Network (RNN) architecture, which excel at capturing temporal dependencies and comprehending intricate patterns in sequences. Leveraging these DL models ensures precise and comprehensive intrusion detection in network traffic data, making them ideal for securing cloud-based systems.

2.4.2.1 Gated recurrent unit (GRU) GRUs, as a variant of RNNs, have demonstrated great potential in cloud intrusion detection by effectively addressing the vanishing gradient problem. Their incorporation of gating mechanisms, such as update and reset gates, enables selective information processing, ensuring the retention and propagation of relevant data. This adaptability significantly enhances the RNN's ability to capture long-term dependencies, which is crucial for identifying abnormal activities. As a result, GRUs have emerged as a powerful addition to the RNN family, offering promising advancements in fortifying cloud intrusion detection mechanisms and safeguarding cloud-based systems against cyber threats.

2.4.2.2 Long short-term memory (LSTM) LSTM has emerged as a powerful solution for addressing the long-term dependency issue encountered by traditional RNNs, especially in the context of intrusion detection. Capturing complex patterns associated with cyber threats is of paramount importance in this domain, and LSTM offers significant advantages in achieving this objective. A key advantage of LSTM is its ability to retain information for an extended period, effectively mitigating the vanishing gradient problem commonly faced by RNNs. This characteristic allows LSTM to remember relevant data over long sequences, enabling the identification of critical dependencies for accurate threat detection. Additionally, LSTM incorporates gating

mechanisms, including an input gate, forget gate, and output gate, which regulate the flow of information within the network. These gates enable LSTM to selectively retain or forget information based on its importance, enhancing its performance in capturing complex patterns and filtering out noise. Moreover, LSTM is capable of handling variable-length sequences, which is highly beneficial in intrusion detection where network traffic or log data may vary in length. This flexibility allows LSTM to adaptively process information without the need for fixed-length inputs or padding, improving efficiency and effectiveness. Furthermore, LSTM exhibits robustness to noisy and incomplete data by selectively focusing on relevant features and filtering out irrelevant or misleading information. This robustness enhances its ability to detect and analyze cyber threats even in the presence of background noise. Finally, LSTM's ability to model long-range dependencies and capture intricate temporal relationships enables the effective identification of sophisticated attack patterns spanning multiple time steps, thereby enhancing the accuracy and efficacy of intrusion detection systems. These advantages position LSTM as a valuable and versatile tool for detecting intrusions, not only in cloud environments but also across diverse computing landscapes.

2.4.2.3 Stacked LSTM Stacked LSTM is an extended version of the LSTM architecture that involves stacking multiple LSTM layers. In this configuration, each layer receives the output from the preceding LSTM layer as its input, resulting in a hierarchical structure. This arrangement facilitates the learning of representations at various levels of abstraction, with each layer extracting and processing distinct features from the input data. The lower layers specialize in capturing short-term dependencies, while the higher layers excel at capturing longer-term dependencies.

By incorporating stacked LSTM layers, the architecture significantly enhances the modeling capacity of the LSTM network, making it well-suited for handling sophisticated intrusion detection tasks. The inclusion of multiple layers enables the model to learn intricate patterns and relationships between different features in the data, leading to improved detection accuracy.

2.4.2.4 Bidirectional LSTM (Bi-LSTM) Bi-LSTM is an advanced variant of the LSTM architecture that enhances the network's ability to capture context from both past and future data points in a sequence. This is achieved by processing the input data in two separate directions: forward and backward. In a Bi-LSTM, two LSTM layers are employed, with one processing the input sequence in its original order and the other processing it in reverse. The outputs of these two layers are then combined to generate the final output.

Table 4 Hyperparameters for the RNN-based Models

Hyperparameters	Values
Epochs	30
Batch size	64
Activation function	tanh
Loss function	Sparse categorical cross-entropy
Optimizer	Adam
Dropout	0.2
Regularizer	L2
Learning Rate	0.001

The primary advantage of Bi-LSTM is its capacity to consider both past and future context when analyzing sequences of network data. This allows the model to better understand the relationships between elements in a sequence and make more accurate predictions about potential intrusions. By leveraging the bidirectional information flow, Bi-LSTM can effectively capture complex patterns and dependencies in network traffic, leading to improved detection of cyber threats.

Incorporating Bi-LSTM into intrusion detection systems can result in more robust and accurate models, capable of identifying a wide range of cyberattacks. The bidirectional nature of Bi-LSTM enables it to outperform traditional LSTM architectures in many cases, making it a valuable addition to the arsenal of techniques used for enhancing network security.

2.4.3 Optimizing hyperparameters for DL models

In the development of our DL models, we meticulously selected and fine-tuned a range of hyperparameters to achieve optimal performance. This subsection provides a comprehensive overview of the specific hyperparameters employed in our models, accompanied by justifications and insights into their functionalities.

Table 4 presents the key hyperparameters used in our DL models, including Stacked LSTM, GRU, Bi-LSTM, and LSTM architectures. The number of parameters in each model varied, and we leveraged regularization techniques such as L2 regularization (with a coefficient of 0.001) to prevent overfitting and enhance generalization.

The activation function played a crucial role in capturing complex patterns and dependencies within the data. For our RNN-based models, the tanh activation function was chosen due to its ability to handle sequential data and learn long-term dependencies effectively. Furthermore, we employed additional components to optimize the model's performance. Dropout layers, with a rate of 0.2, were incorporated to mitigate overfitting, while the softmax activation function

Table 5 ML Models Evaluation for Detecting the Attacks within our Generated Dataset

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
DT	97.26	97.24	97.26	97.25
RF	97.60	97.60	97.60	97.59
XGBoost	97.73	97.72	97.73	97.72
MLP	88.82	88.62	88.82	88.61

facilitated multiclass classification in the final layer of our models.

The choice of optimizer was of paramount importance in achieving stable and efficient training. We selected the widely acclaimed Adam optimizer for its exceptional performance in DL tasks. Its adaptive learning rate and momentum-based updates allowed for efficient convergence and superior model accuracy.

To address the specific characteristics of our datasets, we carefully considered the loss functions. For most models, we utilized the sparse categorical cross-entropy loss, which proved to be effective, especially when dealing with imbalanced data. This loss function accounted for class imbalances and enabled accurate predictions across multiple classes. Moreover, we acknowledge the potential of alternative loss functions, such as focal loss, for achieving higher accuracy in imbalanced datasets.

The models were trained for a fixed number of epochs, which we determined through extensive experimentation. The choice of epoch count, typically set at 30, aimed to strike a balance between convergence and overfitting. By monitoring the models' training and validation performance, we ensured the appropriate duration of training for achieving optimal results.

In addition to the above hyperparameters, we also employed a batch size of 64 for training our models. The batch size determines the number of samples processed in each training iteration. It was chosen based on considerations such as available memory, model complexity, and dataset size. The selected batch size of 64 struck a balance between efficient training and effective utilization of computational resources.

Building upon these optimized hyperparameters, we proceed to delve into the subsequent sections where we provide further insights into the performance and results achieved through their utilization.

2.5 Evaluation

The outcomes of our experiments are presented below, where we compare the effectiveness of each ML/DL model and provide valuable insights into the optimal approach for detecting cloud attacks with minimal error rates. The proposed CADS-ML/DL was evaluated using various metrics, including:

- *Accuracy*: quantifies the proportion of correctly classified instances out of all instances in the dataset. It is a simple and intuitive metric but can be misleading in cases of imbalanced datasets.
- *Precision*: calculates the proportion of true positives out of all predicted positives. It is a useful metric when the focus is on minimizing false positives.
- *Recall*: computes the proportion of true positives out of all actual positives. It is a useful metric when the focus is on minimizing false negatives.
- *F1-score*: a harmonic mean of precision and recall that provides a balanced measure of the classifier's performance.
- *Confusion matrix*: provides a detailed breakdown of the predicted and actual labels for each class. It is a useful tool for analyzing the classifier's performance on specific classes and identifying any imbalances or biases in the datasets.
- *Cross-Entropy Loss Function*: indicates the model's performance in terms of how well it is able to minimize the difference between the predicted output and the actual output during training and validation. Lower loss values indicate better model performance. Cross-entropy is defined as:

$$LCE = -\sum_{(i=1)}^{n} ti \cdot Log(pi) \tag{2}$$

where t_i is the true label and p_i is the Softmax probability for the ith class. Both categorical cross entropy and sparse categorical cross-entropy have the same loss function as defined above. The only difference between the two is in how the labels are represented. Categorical cross-entropy assumes that the labels are one-hot encoded vectors, while sparse categorical cross-entropy assumes that the labels are integers. In this work, we used sparse categorical cross-entropy as our labels were integer encoded.

- *Stratified fivefold cross-validation*: A popular technique used to evaluate the performance of ML and DL models on imbalanced datasets. In this technique, the dataset is first divided into five roughly equal-sized folds, and then the stratified cross-validation is applied. Specifically, each fold is made up of a roughly equal number of samples from each class, and the model is trained on nine folds and tested

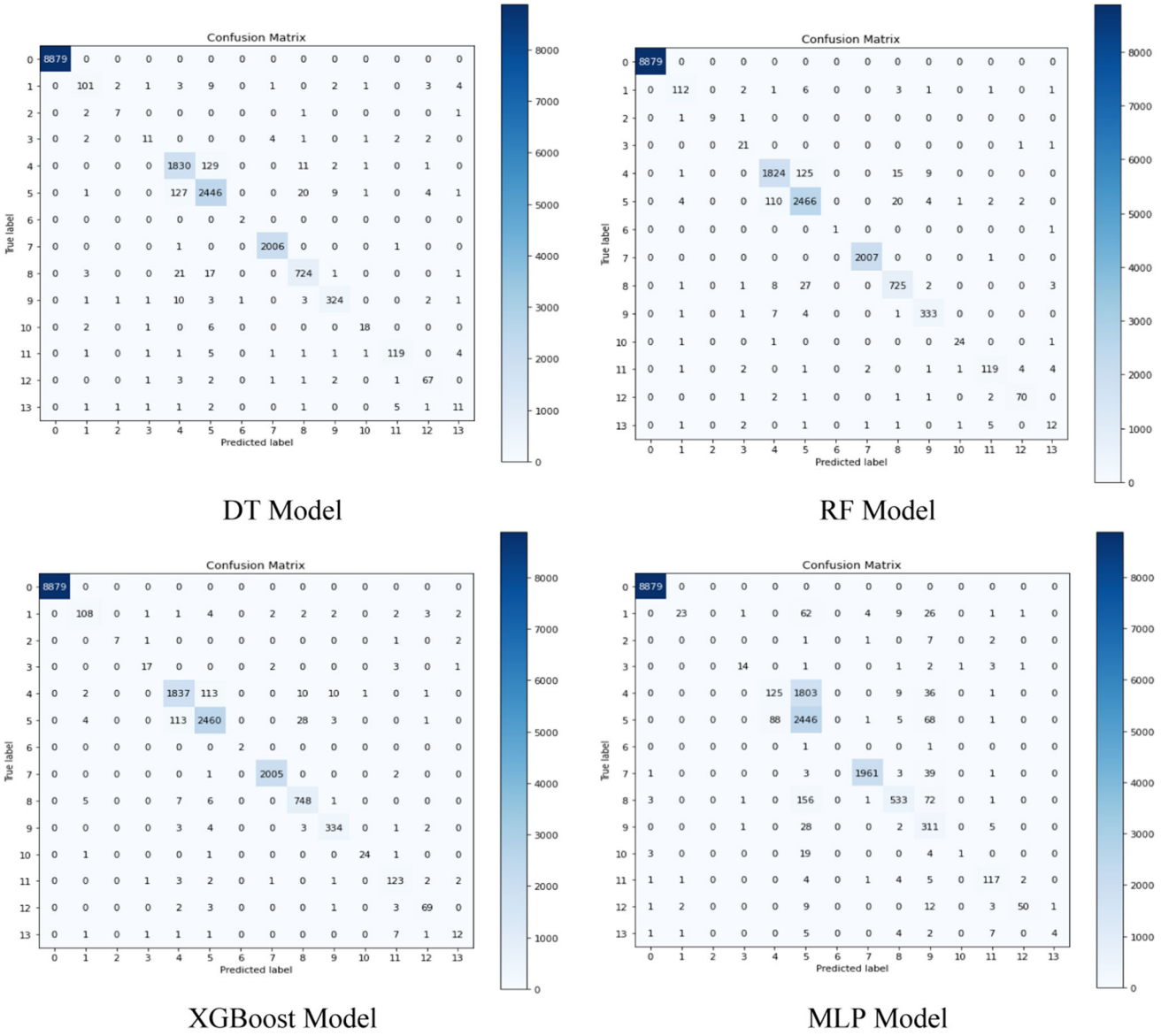


Fig. 14 Confusion matrix of the tested ML models on our generated dataset

on the remaining fold. This process is repeated ten times, with each fold used once for testing and nine times for training. The final evaluation metric is the average of the metrics obtained from each fold. Stratified fivefold cross-validation is a powerful technique as it ensures that the evaluation is done on a representative sample of each class in the dataset and provides a more reliable estimate of the model's true performance.

Based on the evaluation results presented in Table 5 and Fig. 14, it can be observed that the MLP model has the lowest accuracy, precision, recall, and F1-score among the four models, indicating that it is the least effective in detecting cloud attacks. On the other hand, the RF and XGBoost models outperformed the other ML models achieving an accuracy of 97.60% and 97.73%, respectively. These results suggest that XGBoost is the most effective ML model for detecting cloud attacks with minimal error rates.

Looking at the results of Table 6 and Figs. 15, 16, 17, we can see that all four DL models perform fairly well, with accuracy ranging from 88.77% to 89.06%. The GRU model achieved the highest accuracy of 89.06%, closely followed by the LSTM model with 89.02%.

In terms of precision, recall, and F1-score, all four models have similar scores, with only small variations. This suggests that all of the tested models are performing similarly in terms

Table 6 DL Models Evaluation for Detecting the Attacks within our Generated Dataset

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
GRU	89.06	89.00	89.06	88.77
LSTM	89.02	88.99	89.02	88.43
Stacked LSTM	88.81	88.65	88.81	88.61
Bi-LSTM	88.77	88.93	88.77	88.70

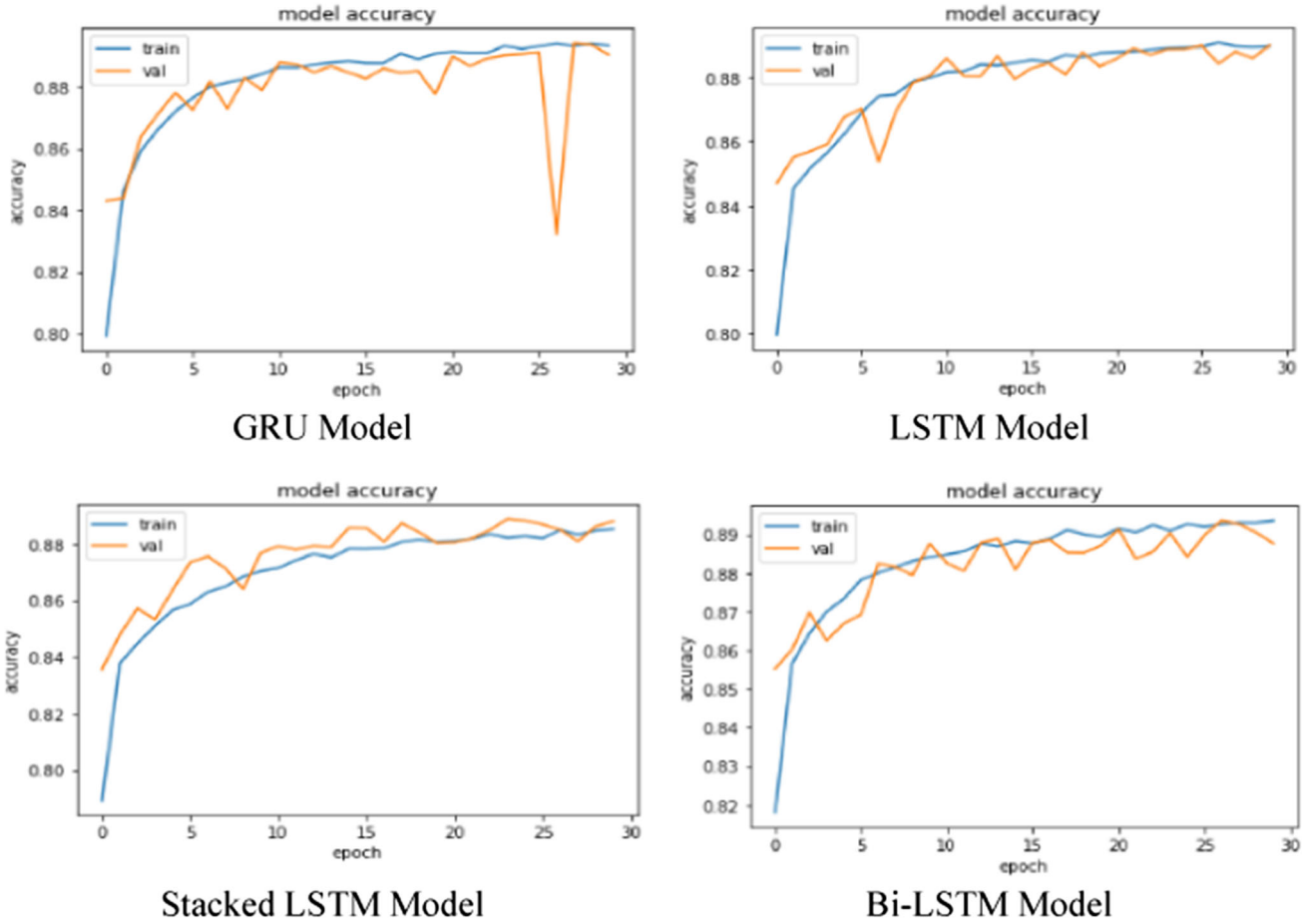


Fig. 15 DL Models Accuracy and Loss on Our Generated Dataset

of their ability to correctly classify instances, with no one model clearly outperforming the others.

Based on the previously discussed results, XGBoost is the optimal approach for detecting cloud attacks, while RF model is alternative option that could also be considered. The detailed results obtained with XGBoost on both our generated dataset and the CSE-CICIDS2018 are summarized in Tables 7 and 8, respectively.

According to Table 7, XGBoost exhibited outstanding accuracy, precision, recall, and F1-score metrics for the vast majority of attack categories with flawless performance for the Benign and SQL Injection. These results highlight the potential effectiveness of XGBoost in detecting various cloud-based attacks and improving the security of cloud computing environments.

Table 8 and Fig. 18 validate the exceptional performance of XGBoost, with almost perfect precision, recall, and F1-score metrics, except for the DDoS HOIC and SQL Injection categories, where the recall rate was slightly lower at 99.93% and 85.71%, respectively. The overall accuracy of the model was an impressive 99.99%, indicating the efficacy of XGBoost in identifying cloud attacks, even with complex and real-world datasets.

We evaluated the XGBoost model using Stratified five-fold cross-validation in terms of accuracy, false error rate, and computation time. To optimize the computation time, we employed parallel processing across multiple CPU cores.

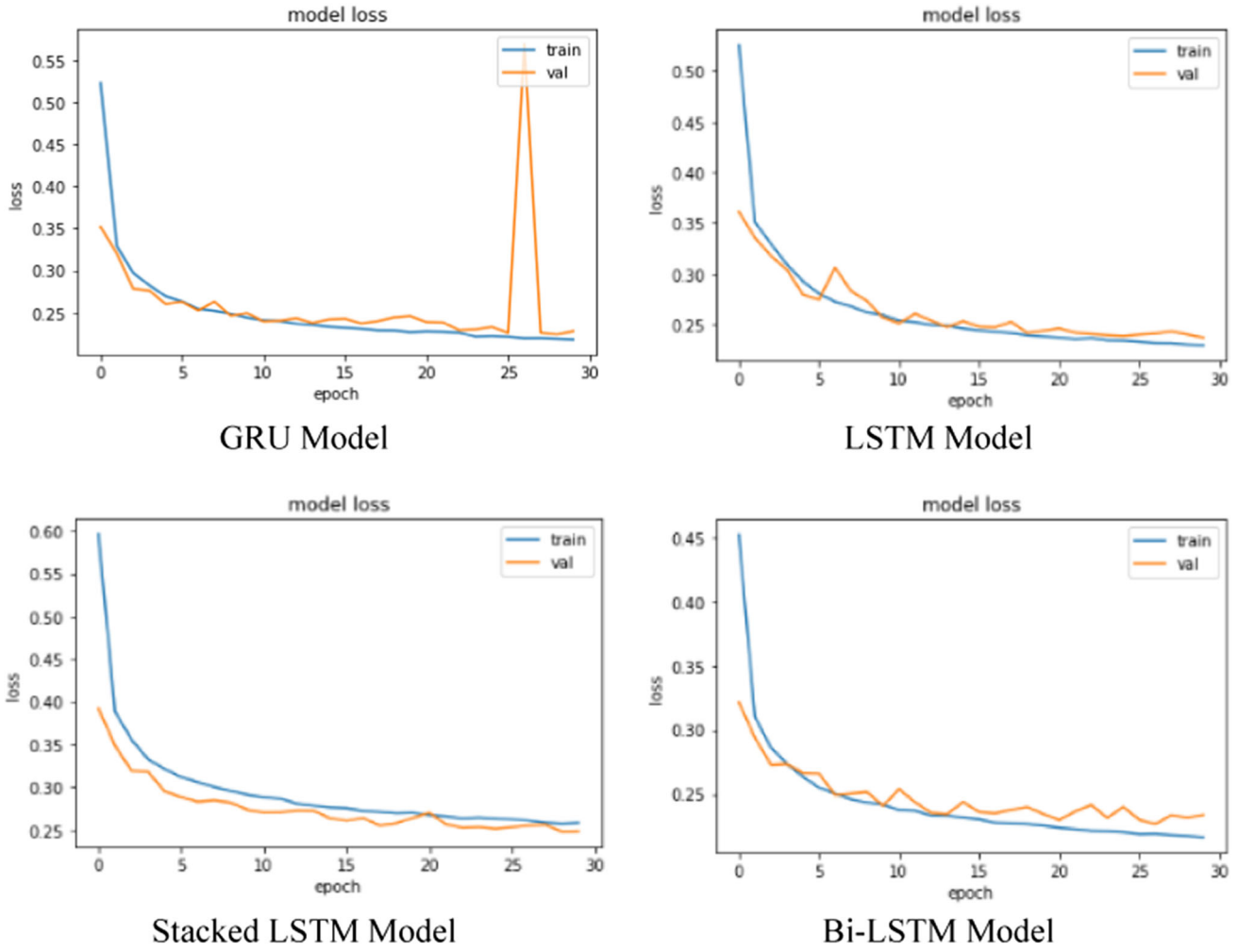


Fig. 16 DL Models Loss on Our Generated Dataset

These techniques can work together to prevent overfitting, improve computation time, and enhance the model's accuracy. The average results are summarized in Table 9 and the detailed accuracy of each fold is depicted in Fig. 19.

According to Table 9, the XGBoost model achieved an average accuracy of 0.9770 on our generated dataset and 0.9999 on CSE-CICIDS2018 dataset. The false error rate was very low for both datasets, at 0.0230 and 0.0001, respectively. However, the computation time was significantly shorter for Our Generated dataset, taking only 9.1242 s, compared to CSE-CICIDS2018, which took 179.0607 s. The results indicate that our proposed system performs exceptionally well, achieving a high level of accuracy and an extremely low false error rate.

Looking at the detailed accuracy of each fold in Fig. 19, we can see that the accuracy for our generated dataset remained relatively consistent across all folds, with a small decrease in fold 4. On the other hand, the accuracy for CSE-CICIDS2018 was consistently high across all folds, except for a slight decrease in fold 4.

These findings imply that the proposed CADS-ML/DL can be applied to various network intrusion detection tasks and may be challenging to improve upon these results.

3 Comparison with related works

This section outlines a comparison of our proposed intrusion detection approach with the most up-to-date proposals, utilizing the CSE-CICIDS2018 dataset. We assess the effectiveness of our CADS-ML/DL using multiple metrics and analyze the findings in Table 10 and Fig. 20.

Our approach achieved the highest accuracy among all compared methods, with the XGBoost model achieving an impressive score of 0.9999. Additionally, our approach demonstrated exceptional precision, recall, and F1-score, all also at 0.9999.

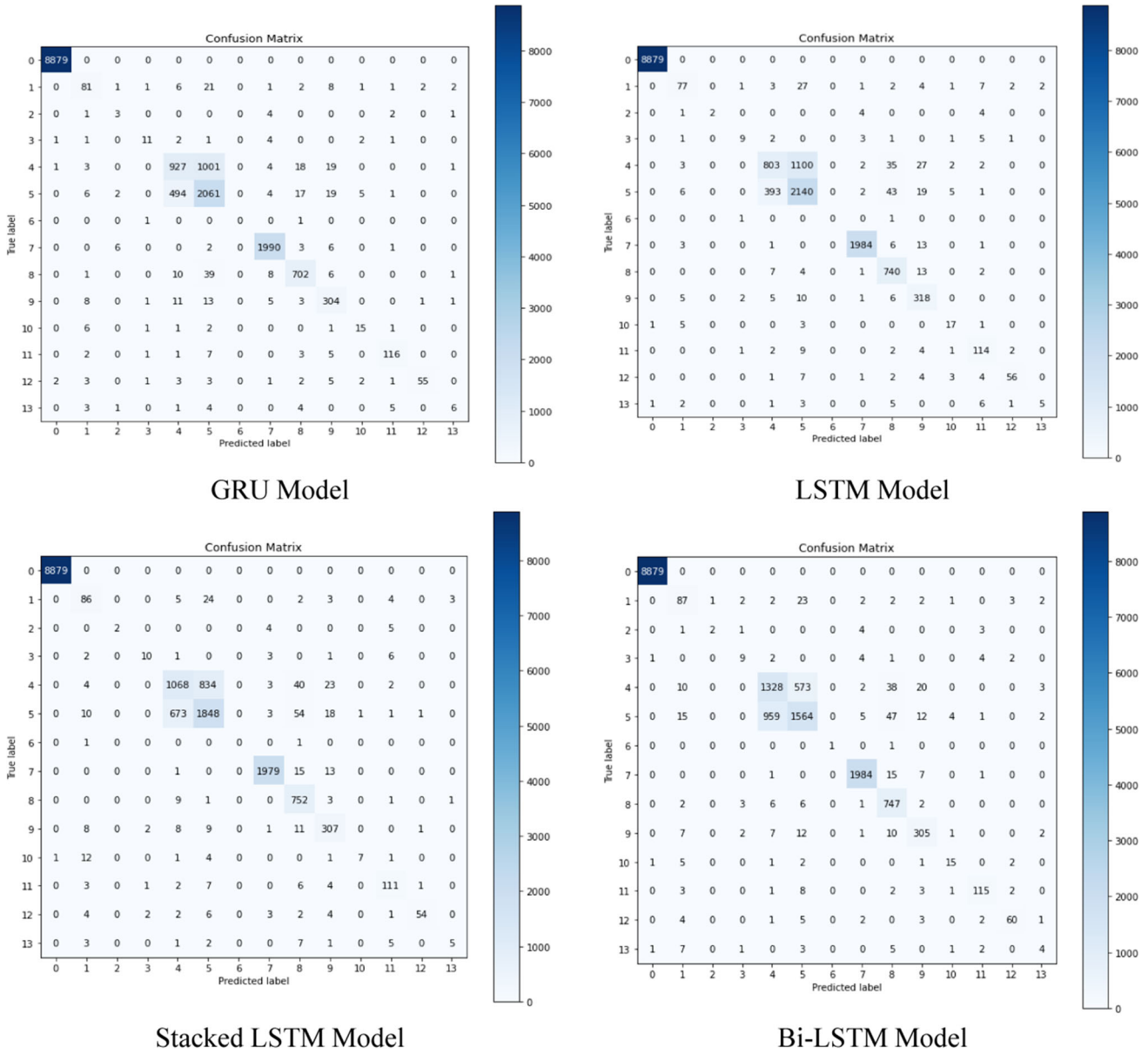


Fig. 17 Confusion matrix of the tested DL models on Our Generated Dataset

Our findings demonstrate that our proposed methodology is highly effective in detecting cloud attacks, surpassing the performance of currently available cutting-edge approaches.

4 Conclusion

Incorporating ML and DL techniques into a cloud-based IDS offers significant potential for organizations seeking to enhance their network security posture. In this research article, we propose CADS-ML/DL, a novel and efficient multi-attack detection system.

CADS-ML/DL demonstrates remarkable performance in identifying various types of network attacks, boasting an impressive average accuracy of 0.9999 and an exceptionally low false error rate of 0.0001 when tested on the CSE-CICIDS2018 dataset.

Table 7 Multi-class classification report of XGBoost on Our Generated Dataset

	Precision (%)	Recall (%)	F1-score (%)	Support
Benign	100	100	100	8879
DDoS Ripper	89.26	85.04	87.10	127
DoS HOIC	100	63.64	77.78	11
DDoS Hammer	80.95	73.91	77.27	23
DoS LOIC	93.39	93.06	93.23	1974
DoS SYNFlood	94.80	94.29	94.54	2609
SQL Injection	100	100	100	2
BruteForce	99.75	99.85	99.80	2008
Scanning	94.56	97.52	96.02	767
DoS Slowloris	94.89	96.25	95.57	347
XSS	96.00	88.89	92.31	27
DDoS HULK	86.01	91.11	88.49	135
Command injection	87.34	88.46	87.90	78
Directory traversal	63.16	50.00	55.81	24
Accuracy (%)			97.73	17,011
Macro average (%)	91.44	87.29	88.99	17,011
Weighted average (%)	97.72	97.73	97.72	17,011

Table 8 Multi-class classification report of XGBoost on CSE-CICIDS2018 Dataset

	Precision (%)	Recall (%)	F1-score (%)	Support
Benign	99.98	100	99.99	208,929
DDoS HOIC	100	99.93	99.96	57,238
Bot	100	100	100	137,202
FTP-BruteForce	100	100	100	346
SSH-Bruteforce	100	100	100	2198
DoS GoldenEye	100	100	100	8302
DoS Slowloris	99.98	100	99.99	38,671
SQL Injection	100	85.71	92.31	7
DDoS LOIC-UDP	100	99.98	99.99	37,518
Accuracy (%)			99.99	490,411
Macro average (%)	99.99	98.40	99.14	490,411
Weighted average (%)	99.99	99.99	99.99	490,411

This cloud-based IDS is designed to be easily deployable and scalable in any cloud environment, making it an ideal solution for organizations of all sizes to effectively protect against a wide range of cyber threats.

However, it is essential to acknowledge that CADS-ML/DL has certain constraints, such as the requirement for labeled training data, the potential for false positives, and the need for ongoing monitoring and maintenance. Therefore, it

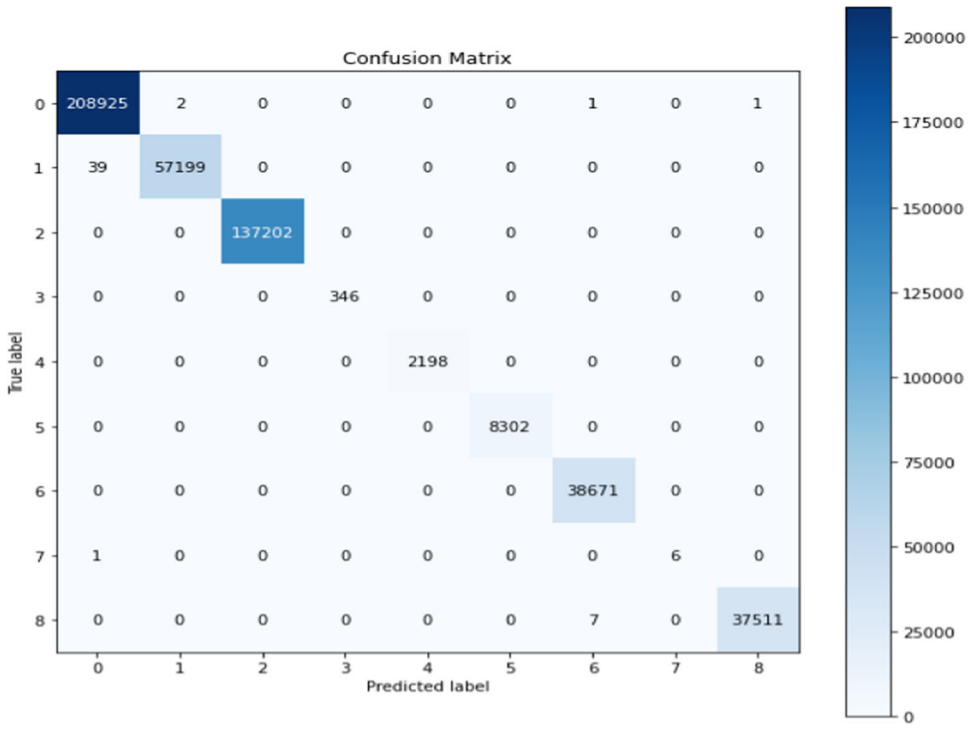


Fig. 18 Confusion matrix of XGBoost Model on CSE-CICIDS2018 Dataset

Table 9 Evaluation of XGBoost Model using Stratified 5-flod cross-validation

Dataset	Accuracy	False error rate	Computation time (s)
Our generated dataset	0.9770	0.0230	9.1242
CSE-CICIDS2018	0.9999	0.0001	179.0607

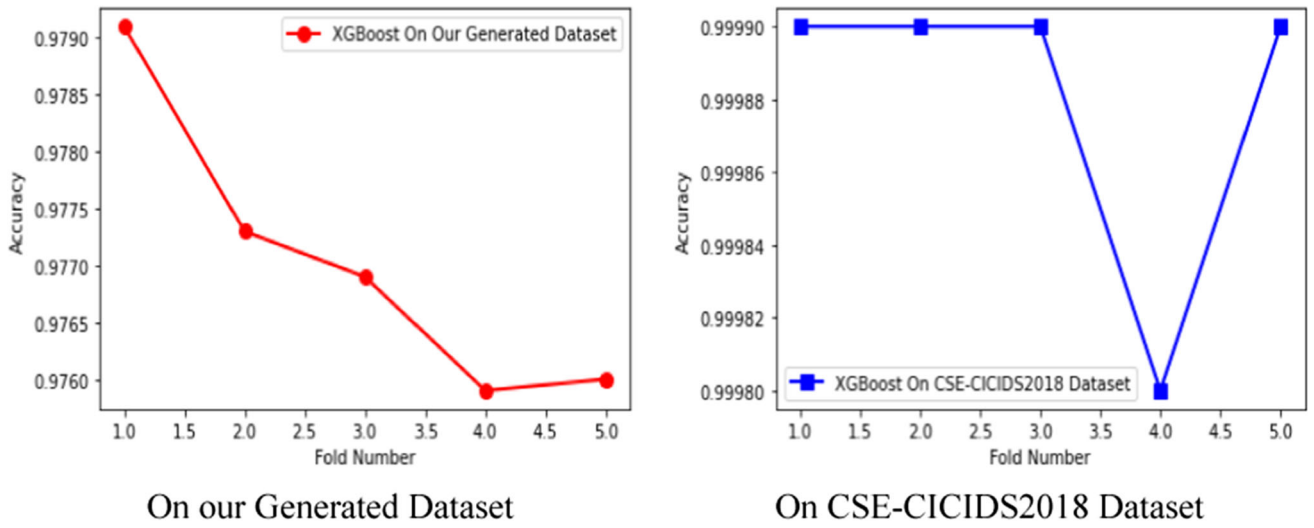


Fig. 19 Evaluation of XGBoost Model using Stratified 5-flod cross-validation

Table 10 Overall performance comparison to existing Approaches on CSE-CICIDS2018

Authors	Algorithms	Precision	Recall	F1-score
Karatas et al. [23]	Adaboost	0.9970	0.9969	0.9970
Seth et al. [27]	Light GBM	0.9933	0.9606	0.9757
Wang et al. [31]	CNN + LSTM	0.9843	0.9884	0.9843
Our approach	XGBoost	0.9999	0.9999	0.9999

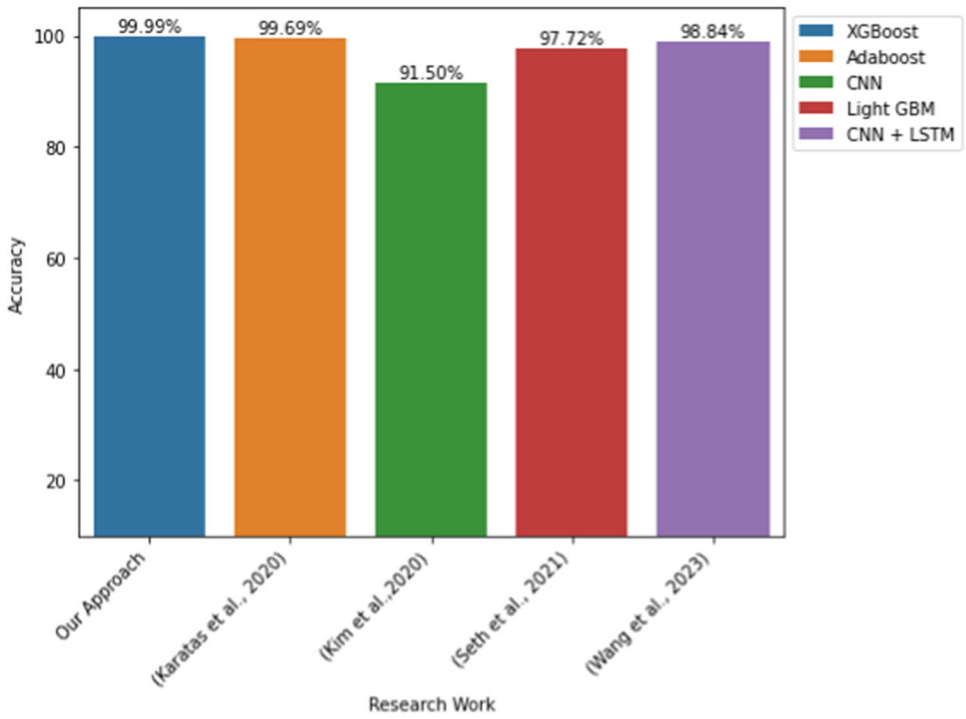


Fig. 20 Accuracy comparison with recent approaches

should be used in conjunction with existing security measures, and regular updates and fine-tuning should be applied to ensure the system is always up-to-date and accurate in detecting the latest attack tactics.

Author contributions AB, CD, EF, and GH conceived and planned the experiments. AB carried out the experiments. AB, CD, EF, and GH contributed to the interpretation of the results. AB took the lead in writing the manuscript. All authors provided critical feedback and helped shape the research, analysis, and manuscript.

Data availability The datasets generated and analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Jathanna, R., Jagli, D.: Cloud computing and security issues. Int. J. Eng. Res. Appl. **07**, 31–38 (2017). https://doi.org/10.9790/9622-0706053138
2. Subramanian, N., Jeyaraj, A.: Recent security challenges in cloud computing. Comput. Electr. Eng. **71**, 28–42 (2018). https://doi.org/10.1016/j.compeleceng.2018.06.006
3. Almudaires, Fajer & Almaiah, Mohammed & Almaayah, Drmohammed. (2021). Data an Overview of Cybersecurity Threats on Credit Card Companies and Credit Card Risk Mitigation. pp 732–738. https://doi.org/10.1109/ICIT52682.2021.9491114.
4. Ahmad, W., Rasool, A., Javed, A.R., Baker, T., Jalil, Z.: Cyber security in IoT-based cloud computing: a comprehensive survey. Electronics **11**(1), 16 (2022). https://doi.org/10.3390/electronics11010016
5. Alawida, M., Omolara, A.E., Abiodun, O.I., Al-Rajab, M.: A deeper look into cybersecurity issues in the wake of Covid-19: a survey. J. King Saud Univ. Comput. Inf. Sci. **34**(10), 8176–8206 (2022). https://doi.org/10.1016/j.jksuci.2022.08.003
6. Shaikh, F.A., Siponen, M.: Information security risk assessments following cybersecurity breaches: The mediating role of top management attention to cybersecurity. Comput. Security **124**, 102974 (2023). https://doi.org/10.1016/j.cose.2022.102974
7. Abdullayeva, F.J.: Advanced persistent threat attack detection method in cloud computing based on autoencoder and softmax

regression algorithm. Array **10**, 100067 (2021). https://doi.org/10.1016/j.array.2021.100067
8. Abdulsalam, Y.S., Hedabou, M.: Security and privacy in cloud computing: technical review. Future Internet. **14**(1), 11 (2022). https://doi.org/10.3390/fi14010011
9. Golightly, L., Chang, V., Xu, Q.A., Gao, X., Liu, B.S.: Adoption of cloud computing as innovation in the organization. Int. J. Eng. Bus. Manag. **14**, 18479790221093990 (2022). https://doi.org/10.1177/18479790221093992
10. Rana, P., Batra, I., Malik, A., Imoize, A.L., Kim, Y., Pani, S.K., Goyal, N., Kumar, A., Rho, S.: Intrusion detection systems in cloud computing paradigm: analysis and overview. Complexity **3999039**, 14 (2022). https://doi.org/10.1155/2022/3999039
11. Azab, A., Khasawneh, M., Alrabaee, S., Raymond Choo, K.-K., Sarsour, M.: Network traffic classification: techniques datasets and challenges. Digital Commun. Netw. (2022). https://doi.org/10.1016/j.dcan.2022.09.009
12. Zhang, Y., Liu, Y., Guo, X., Liu, Z., Zhang, X., Liang, K.: A BiLSTM-based DDoS attack detection method for edge computing. Energies **15**(21), 7882 (2022). https://doi.org/10.3390/en15217882
13. Patel, A., Taghavi, M., Bakhtiyari, K., et al.: An intrusion detection and prevention system in cloud computing: a systematic review. J. Netw. Comput. Appl. **36**(1), 25–41 (2013). https://doi.org/10.1016/j.jnca.2012.08.007
14. Mamaheswari, K., Sujatha, S.: Impregnable defence architecture using dynamic correlation-based graded intrusion detection system for cloud. Defence Sci. J. **67**, 645–653 (2017). https://doi.org/10.14429/dsj.67.11118
15. Iqbal, Farkhund & Batool, Rabia & Fung, Benjamin & Aleem, Saiqa & Abbasi, Ahmed & Javed, Abdul Rehman. (2021). Tweet-to-act: towards tweet-mining framework for extracting terrorist attack-related information and reporting. IEEE access. PP. 1–1. https://doi.org/10.1109/ACCESS.2021.3102040.
16. Díaz-Verdejo, J., Muñoz-Calle, J., Estepa Alonso, A., Estepa Alonso, R., Madinabeitia, G.: On the detection capabilities of signature-based intrusion detection systems in the context of web attacks. Appl. Sci. **12**(2), 852 (2022). https://doi.org/10.3390/app12020852
17. Cebi, C., Bulut, F., Firat, H., Sahingoz, O., Baydogmus, K., Gozde.: Deep learning based security management of information systems: a comparative study. J. Adv. Inf. Technol. (2020). https://doi.org/10.12720/jait.11.3.135-142
18. Atefinia, R., Ahmadi, M.: Network intrusion detection using multi-architectural modular deep neural network. J. Supercomput. **77**, 3571–3593 (2021). https://doi.org/10.1007/s11227-020-03410-y
19. Aldallal, A.: Toward efficient intrusion detection system using hybrid deep learning approach. Symmetry. **14**(9), 1916 (2022). https://doi.org/10.3390/sym14091916
20. Balasubramaniam, S., Vijesh Joe, C., Sivakumar, T.A., Prasanth, A., Satheesh Kumar, K., Kavitha, V., Dhanaraj, R.K.: Optimization enabled deep learning-based DDoS attack detection in cloud computing. Int. J. Intell. Syst. **2039217**, 16 (2023). https://doi.org/10.1155/2023/2039217
21. Talpur, N., Abdulkadir, S.J., Alhussian, H., Hasan, M.H., Aziz, N., Bamhdi, A.: A comprehensive review of deep neuro-fuzzy system architectures and their optimization methods. Neural Comput. & Appl. **34**, 1837–1875 (2022). https://doi.org/10.1007/s00521-021-06807-9
22. Talpur, N., Abdulkadir, S.J., Alhussian, H., Hasan, M.H., Aziz, N., Bamhdi, A.: Deep neuro-fuzzy system application trends, challenges, and future perspectives: a systematic survey. Artif. Intell. Rev. **13**, 1–49 (2023). https://doi.org/10.1007/s10462-022-10188-3
23. Karatas, G., Demir, O., Sahingoz, O.K.: Increasing the performance of machine learning-based IDSs on an imbalanced and up-to-date dataset. IEEE Access **8**, 32150–32162 (2020). https://doi.org/10.1109/ACCESS.2020.2973219
24. Zhou, Y., Cheng, G., Jiang, S., Dai, M.: Building an efficient intrusion detection system based on feature selection and ensemble classifier. Comput. Netw. **174**, 107247 (2020). https://doi.org/10.1016/j.comnet.2020.107247
25. Kim, J., Kim, J., Kim, H., Shim, M., Choi, E.: CNN-based network intrusion detection against denial-of-service attacks. Electronics **9**(6), 916 (2020). https://doi.org/10.3390/electronics9060916
26. Rehman, Su., Khaliq, M., Imtiaz, S.I., Rasool, A., Shafiq, M., Javed, A.R., Jalil, Z., Bashir, A.K.: DIDDOS: an approach for detection and identification of Distributed Denial of Service (DDoS) cyberattacks using Gated Recurrent Units (GRU). Future Gener. Comput. Syst. **118**, 453–466 (2021). https://doi.org/10.1016/j.future.2021.01.022
27. Seth, S., Singh, G., Kaur Chahal, K.: A novel time efficient learning-based approach for smart intrusion detection system. J. Big Data **8**, 111 (2021). https://doi.org/10.1186/s40537-021-00498-8
28. Fu, Y., Du, Y., Cao, Z., Li, Q., Xiang, W.: A deep learning model for network intrusion detection with imbalanced data. Electronics **11**(6), 898 (2022). https://doi.org/10.3390/electronics11060898
29. Sydney Mambwe Kasongo: A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework. Comput. Commun. **199**, 113–125 (2023). https://doi.org/10.1016/j.comcom.2022.12.010
30. Abdelkhalek, A., Mashaly, M.: Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning. J. Supercomput. (2023). https://doi.org/10.1007/s11227-023-05073-x
31. Wang, Y.-C., Houng, Y.-C., Chen, H.-X., Tseng, S.-M.: Network anomaly intrusion detection based on deep learning approach. Sensors **23**(4), 2171 (2023). https://doi.org/10.3390/s23042171
32. A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018) was accessed on 02/01/2023 from https://registry.opendata.aws/cse-cic-ids2018.
33. Khan, M.A.: HCRNNIDS: hybrid convolutional recurrent neural network-based network intrusion detection system. Processes **9**(5), 834 (2021). https://doi.org/10.3390/pr9050834