IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 1 October 2023, accepted 15 October 2023, date of publication 19 October 2023, date of current version 25 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3325992

RESEARCH ARTICLE

SecFedIDM-V1: A Secure Federated Intrusion Detection Model With Blockchain and Deep Bidirectional Long Short-Term Memory Network

EMMANUEL BALDWIN MBAYA[1,2,3], EMMANUEL ADETIBA[1,2,4], (Member, IEEE), JOKE A. BADEJO[1,2], (Member, IEEE), JOHN SIMON WEJIN[1,2], OLUWADAMILOLA OSHIN[1,2], (Member, IEEE), OLISAEMEKA ISIFE[1], SURENDRA COLIN THAKUR[5], SIBUSISO MOYO[6], AND EZEKIEL F. ADEBIYI[2]

[1]Department of Electrical and Information Engineering, Covenant University, Ota, Ogun State 112104, Nigeria
[2]Covenant Applied Informatics and Communication African Center of Excellence, Covenant University, Ota, Ogun State 112104, Nigeria
[3]Department of Computer Science, Federal University, Gashua, Yobe 631001, Nigeria
[4]HRA, Institute for Systems Science, Durban University of Technology, Durban 4001, South Africa
[5]Department of Information Technology and KZN e-Skills Co-Laboratory, Durban University of Technology, Durban 4001, South Africa
[6]Department of Mathematical Sciences, School for Data Science and Computational Thinking, Stellenbosch University, Stellenbosch 7602, South Africa

Corresponding author: Emmanuel Adetiba (emmanuel.adetiba@covenantuniversity.edu.ng)

This work was supported in part by the Covenant Applied Informatics and Communication Africa Centre of Excellence (CApIC-ACE), domiciled at Covenant University with the ACE Impact Grant from the World Bank through the National University Commission, Nigeria; and in part by the Covenant University Center for Research, Innovation, and Discovery (CUCRID), Covenant University. The work of Emmanuel Adetiba was supported in part by the Google TensorFlow College Outreach Award through a machine learning workshop session based on this project.

ABSTRACT Cloud computing is a technology for efficiently using computing infrastructures and a business model for selling computing resources and services. However, intruders find such complex and distributed infrastructures appealing targets for cyber-attacks. Cyber-attacks are severe threats that can jeopardize the quality of service provided to clients and compromise data integrity, confidentiality, and availability. Cyber-attacks are becoming more complex, making it more challenging to detect intrusions effectively. Due to the high traffic and increased malicious activities on the Internet, a single Intrusion Detection System (IDS) can be overwhelmed. Despite the various Deep Learning (DL) approaches that have been proposed as alternative solutions, there are still pertinent security issues to be addressed especially in federated cloud computing domains. This work proposes a Secure Federated Intrusion Detection Model Version 1 (SecFedIDM-V1) using blockchain technology and Bidirectional Long Short-Term Memory (BiLSTM) Recurrent Neural Network (RNN). The Cobourg Intrusion Detection Dataset (CIDDS) was acquired, pre-processed and split into 60:20:20, 70:15:15, and 80:10:10 for training, testing, and validation respectively to develop the proposed intrusion traffic classification component of the proposed model. The developed SecFedIDM-V1 was later deployed as a Python-based web application that captures network packets for classifying attacks into normal or an attack type. The attack packets are recorded in a Hyperledger Fabric (a private blockchain technology) to serve as a signature database to be used by other nodes in the network. From the evaluation results of the intrusion classifier, the 80:10:10 BiLSTM network performed better than GRU with a Precision of 0.99624, Recall of 0.99906, F1 Score of 0.99614, False Positive Rate (FPR) of 0.00094, False Negative Rate (FNR) of 0.00395 and True Positive Rate (TPR) of 0.99605. The SecFedIDM-V1 can be deployed alongside Firewalls in a federated cloud computing environment to reinforce the security of the infrastructure.

INDEX TERMS Blockchain, intrusion detection, deep learning, recurrent neural network.

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

I. INTRODUCTION

Cloud computing has emerged to answer the quest for the provision of computation as a utility like electricity and water. Cloud computing provides IT-based services to geographically dispersed locations via the Internet. In this approach, users utilize the computational power provided by the Cloud Service Providers (CSP) and pay only for what they consume as opposed to paying an amount upfront or making long–term contracts. By paying for consumed computation power, spending on capital goods could be converted into other operational expenses [1]. Virtually shared resources in the cloud computing domain include computation and storage resources, software applications, operating systems, and network infrastructures. Businesses and individuals gain access to cloud services either through social media, email, web application hosting, etc. Cloud storage solutions like AWS, Microsoft Azure Blobs, and Google Cloud Storage are widely used by different cloud retailers and accessed from multiple devices [2], [3]. Well over $1 trillion has already been devoted actively or passively to cloud-based computing [4].

The benefits of cloud computing technology include extensibility, resource allocation based on need, less management overhead, an adjustable pricing system (pay-as-you-go), and easier application development and delivery. Large data centres come with astronomical upfront expenses. However, smaller businesses can attempt to approach this scale impact by creating federations of computing and storage utilities [5]. Resource migration, resource redundancy, and the combination of complementary resources or services are the three main interoperability elements supported by cloud federation, which consists of services from several providers aggregated in a single pool [6].

The primary cloud service models are Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). IaaS provides virtualized computing, storage, and networking resources. Platform-as-a-Service provides the environment to design, develop, deploy and manage software applications whereas SaaS provides end users with applications without physical machine installation [2], [4], [7], [8]. As technology advances, the number of hacking incidents is on the rise. Security issues have become a major challenge in a complicated technical world. Cloud providers and users report attacks regularly [9]. Detecting cyber threats in the cloud is expensive and time-consuming as they become increasingly sophisticated. In order to prevent and secure Information Technology (IT) cloud infrastructure and operations, active Intrusion Detection Systems or Models (IDS/M) are essential [2].

Intrusion is any unauthorised activity that causes harm to a computer system/network and has the potential to compromise confidentiality, integrity, or availability of information [10]. Intrusion detection models are usually embedded in software applications to check networks or computers for harmful activities to maintain system security [10]. As a preventive measure, IDM identifies and stops potential threats to a system or network before they may cause damage, including both malicious insiders and hackers from the outside [11], [12]. Various experts have proposed using Machine Learning (ML) algorithms to develop IDMs to improve the security of computation and network resources. Machine Learning emulates human cognitive abilities through modelling deduction, inference, extrapolation, and synthesis. It could be leveraged to build IDMs thereby enabling precise identification of malicious traffic, leading to fewer false positive alerts. Common ML algorithms that have been used for developing IDMs include Support Vector Machines (SVM), Decision Trees (DT), Bayesian Networks, Naïve Bayes etc. [13], [14].

However, the foregoing traditional ML methods focus on the manual engineering of features and are less efficient in the presence of enormous data that needs classification in a production environment. Multiple classification tasks reduce accuracy as dataset size grows. In order to overcome the high data analysis challenges of traditional ML algorithms, Deep Learning(DL) algorithms are employed to improve IDMs performance, especially in multi-class scenarios [15].

Lately, blockchain technology has permeated every facet of Information and Communication Technology (ICT), and its usage has increased. Significant increases in the value of cryptocurrencies and notable venture capital investments in blockchain start-up companies have fueled interest in the development of this technology [16], [17]. Primarily, blockchain employs the function of a ledger where all transactions carried out by participating peers are recorded and shared with all participants. With the inclusion of new blocks over time, this form of blockchain continues to expand. Popular cryptocurrency blockchains are publicly available, and transactions may be queried via web platforms such as blockchain.com, allowing anybody to query the blockchain transactions. Blockchain enables parties to do business without a (trusted) third party [17], [18].

This paper proposes a Secure Federated Intrusion Detection Model (SecFedIDM-V1). The model uses Bidirectional Long Short-Term Memory (BiLSTM), a deep learning algorithm, to detect and classify malicious activities on the network by monitoring incoming network packets. The details of the packets classified as malicious are stored in a blockchain ledger in a private network where only certified nodes can access it, and the ledger serves as a signature database. The rest of this paper is organized as follows. Section II provides fundamental understanding of Intrusion Detection Systems (IDSs) as well as machine learning. Related research works on IDS, and the use of intelligent algorithms in IDSs are presented in section II. The design and implementation of the proposed model are discussed in section III. Section IV presents the experimental evaluation results, while the conclusion and future works are presented in section V.

II. BACKGROUND

A. AN OVERVIEW OF INTRUSION DETECTION SYSTEMS

System intrusion is any illegal attempt to gain access to a network, cloud, desktop/laptop or Virtual Machine (VM) with the intent to violate and compromise it, thereby making it vulnerable and permitting unauthorized access. This can be done by exploring vulnerabilities in the system and circumventing or disabling components of the system [19]. An IDS is a security tool that collects and analyses the traffic coming into a network, logs the activity, scans a host system or network for suspicious activity, and alerts system or network administrators. Common examples of IDS include host-based, network-based, and hypervisor-based IDS. Host-based Intrusion Detection Systems (HIDS) monitor physical and virtual hosts individually. When suspicious activity is detected on a host or VM, a warning alert is sent to the user. Some examples of suspicious activities include altering or deletion of system files, a series of inappropriate system operations, or undesirable configuration changes. Network-based Intrusion Detection Systems (NIDS) are typically located at points within the network, like gateways and routers, to monitor anomalies in network traffic as they form the frontlines of network security [9], [20].

Other types of IDS include Signature IDS (SIDS) and Anomaly IDS (AIDS). Signature IDS, (also known as Knowledge-based Detection (KBD) or Misuse Detection (MD)), uses patterns of similarity to identify malicious activities [13], [21]. If a similar pattern is found in the previous records stored in the signature database, an alarm is triggered, notifying the admin that an intrusion has been found [10]. For instance, to circumvent a signature that searches for the filename ''malware.exe,'' an attacker could create a malicious program named ''malware1.exe''. However, anomaly-based detection is capable of identifying unknown attacks. System/network events can be monitored for an extended period to build a baseline profile for abnormal behaviour. Static and dynamic profiles can both be used in practice. Dynamic profiles are updated regularly, whereas static profiles are not. Although AIDS can detect wider classes of intrusions, anomaly-based detection suffers from high false-positive rates [22].

Based on the type of attack response, IDS can be further categorized into active and passive IDSs. It is considered a passive approach when the network or security administrator gets a notification, and the event is logged. Active IDSs, also known as intrusion detection and prevention systems, can identify vulnerabilities, flag suspicious activity, and take preventive action utilizing a variety of response mechanisms to keep up with the development of computer-related crimes. Regarding the mode of operation, IDS schemes can be further divided into online and offline schemes. An online IDS scheme intercepts packets from the network to monitor and handle intrusions. Its performance depends on the number of features to be analyzed. However, offline IDSs can help process stored logs (such as benchmark records), detect attacks, understand them, and mitigate future attacks. In addition, depending on the time the detection was executed, IDS can be categorized as an interval-based or real-time solution [23].

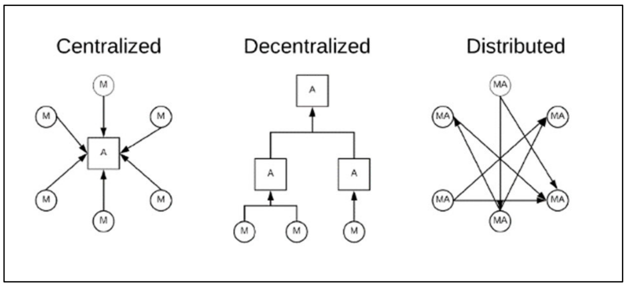


FIGURE 1. Overview of CIDS types.

Collaborative Intrusion Detection Systems/Networks (CIDSs/CIDNs) are intended to improve the detection capabilities of an IDS by permitting single nodes to collect, communicate and share crucial information with other nodes. A CIDS is required because a single node IDS cannot effectively identify sophisticated and complicated intrusions in modern systems. It enables all IDSs to interconnect and collaborate on emerging cyber issues, immunizing them and averting threats from increasing [24], [25]. For instance, a server acting as a central store of logs is more sensitive to network irregularities than an individual IDS since it aggregates traffic characteristics from several detection sensors. Collaborative IDS efficiently identifies widespread attacks in vast networks by providing a global view of the assaults. Integrating different IDSs, CIDS reduces the computational cost of detecting intrusions in various servers and improves detection on the cloud [24], [26]. Due to better detection performance, collaborative intrusion detection frameworks are now widely utilized in enterprises. However, two fundamental challenges, namely data management and trust calculation, remain unresolved [22], [27]. Collaborative IDS are classified, as depicted in Fig. 1, into i) centralized, ii) decentralized/hierarchical and iii) federated/distributed, respectively [26], [27], [28].

The centralized CIDS uses a central server to store logs from all participating units and runs the analysis for detection of an intrusion on the network. This CIDS is liable to a Single Point of Failure (SPoF) and unscalable. In distributed/hierarchical CIDS, the flow is in a hierarchical pattern. The monitoring nodes send data to an analyzing unit; then, from the analyzing unit, data is sent to another unit to identify intrusions in the CIDS. Though the hierarchical CIDS approach is less prone to SPoF when compared to the centralized CIDS, it is also not fully scalable. In distributed/federated CIDS, a peer–peer approach where each node performs the role of an analyzing unit, and a monitoring unit is employed. Distributed/Federated CIDS, as shown in Fig. 1, is very scalable and resistant to SPoF. This research focuses on a distributed (federated) architectural approach to addressing the issues of CIDS.

Data sharing is a significant problem for collaborative detection since not all organizations wish to disclose their data directly. Intrusion detection frequently uses machine learning techniques to create typical profiles, which necessitates many training samples for a classifier. Some firms are unwilling to disclose their data due to privacy concerns, making it challenging to improve detection performance. Secondly, insider threats pose a significant obstacle to coordinated detection and significantly compromise network security. In a distributed and collaborative setting, determining the integrity of an IDS node might be difficult. For instance, it is not easy to quantify the reputation levels of multiple collaborating parties adequately [29].

The issue of privacy and safety of records can be solved with blockchain. With blockchain, peers can share data over a decentralized ledger network. In addition, validators (typically miners) are used to replace third parties and carry out decentralised transaction validation. Distributed consensus is the ability of a group of individuals who do not trust one another to reach an agreement on how to verify a certain quantity of a digital transaction without the authorization of a reliable third party [17], [30], [31].

The benefits of blockchain, a novel distributed ledger system, include data security, autonomy, openness, privacy, immutability, efficiency, speed, and cost reduction. The eviction of central authority and the facilitation of the development of autonomous, secure, and transparent systems result from generating trust across non-trusting organizations. Several organizations, consortiums, and nations are integrating blockchain technology into their systems to reap its benefits [32], [33]. Blockchain is a reliable way to maintain an unalterable distributed ledger. It has great potential in the cloud computing industry, particularly for protecting the provenance of data items across cloud infrastructure. The data's ancestry is determined by the data's provenance, based on the data object's real origination. Assume that authentic data provenance exists in the cloud for all cloud-stored data, distributed data calculations, and data transactions, it would be capable of detecting insider attacks, and identifying a reliable source of system or network breaches [34].

Individual IDS alerts are recorded as transactions in the blockchain. Before adding a transaction to the blockchain, each unique IDS connecting to the centralized server runs a consensus procedure to validate the transaction. This architecture ensures that only verified, validated alerts are updated on the centralized server. These notifications are tamper-resistant and accessible to all IDS node participants [35]. No block is appended to the blockchain without the consensus of all network participants. Due to the network's transparency, attacks such as man-in-the-middle and network session hijacking are impossible. The data included in a block cannot be modified retrospectively without modifying the following block. As depicted in Fig. 2, since all participants share the identical blockchain, any effort to alter the data of a node is rejected during the ratification of the block, making it an integrated system [25], [36].

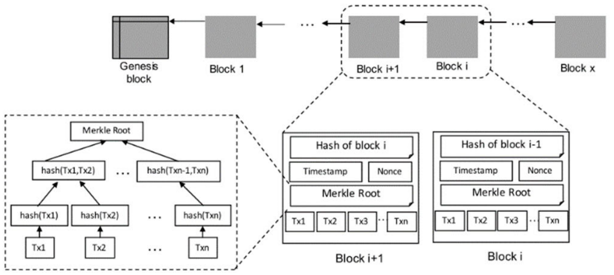


FIGURE 2. An in-depth view of a conventional blockchain.

B. INTRUSION DETECTION DATASET

The choice of dataset is one of the most vital aspects of developing an IDS. The dataset is used in modelling an IDS during the training phase and subsequently employed to evaluate the IDS's model performance. Setting up and capturing real-time attacks from regular network traffic is quite expensive and technically demanding. Thus, authors have resorted to using publicly accessible benchmark datasets from different research laboratories. Common datasets employed for IDS studies include Knowledge Discovery in Databases (KDD) Cup 99, Canadian Institute for Cybersecurity IDS 2017 (CIC-IDS2017), and Australian Defense Force Academy (ADFA) [10], [37]. The Department of Defense Advanced Research Projects Agency (DARPA) curated the Knowledge Discovery and Data Mining (KDD98) dataset in 1998 as the first attempt to create an IDS dataset. The dataset contains information about TCP sessions that begin and end at predetermined intervals and simulates a wider range of attacks on a military network [10], [35]. Despite its importance to IDS research, this dataset has been widely criticized for its inaccuracy and inability to consider real-world situations. This made models trained with the dataset perform poorly during the evaluation phase. The shortcomings in KDD98 led to the generation of a new dataset called the Network Security Laboratory (NSL)-KDD. The NSL-KDD, as a modified and upgraded version of KDD98, contains four categories of attacks. These include Probe, User to Root (U2R), Denial of Service (DoS), and Remote to Local (R2L) [39]. The CIC-IDS2017 dataset comprises of many updated attack cases and satisfies the criteria for actual attacks. The dataset consists of five files containing network traffic information collected over five days. The entirety of the CIC-IDS2017 dataset consists of 3,119,345 records and 83 attributes with 15 class labels (1 normal and 14 attack labels) [40]. The CIC-IDS2017 dataset contains up-to-date flow-based network data of normal and attack traffic. This dataset is suitable for extensive NIDS cloud benchmarking [41], [42]. The Coburg Intrusion Detection Data Sets (CIDDS) [41], [42] were generated on an OpenStack platform. It contains 25,112,036 normal and attack flows with 14 different attributes of unidirectional NetFlow. This dataset was appositely adopted for this study. Further details on the dataset are presented in section III-A.

C. MACHINE LEARNING-BASED INTRUSION DETECTION SYSTEMS

Over the years, statistical approaches, expert techniques, and time-series approaches have been explored to develop models for IDS. However, with the increased data traffic in modern networks, it becomes challenging for IDSs developed using the aforementioned approaches to draw a clear line between legitimate and harmful operations. This drawback necessitated machine learning and deep learning approaches [43], [44]. The rapid development of deep learning theory and technology over the past few years has ushered in a new era of artificial intelligence and provided a novel approach to creating intelligent intrusion detection technologies [15]. A typical example of learning approaches for improving the performance of IDSs is the use of Recurrent Neural Networks (RNNs), which have existed for a couple of years, but whose full potential was only recently begun to be recognised. An RNN comprises input, output, and hidden units. The hidden unit carries out most of the learning computations. Information usually flows from the preceding historical concealment unit to the current timing concealment unit. The RNN model effectively has a one-way data flow from input to hidden nodes. Hidden units are considered the network's memory which stores end-to-end information.

When learning long-term dependency is a model priority, using Gated Recurrent Unit (GRU) networks (a variant of RNN) as a learning technique is more suitable. A GRU network avoids the vanishing and explosion of gradient problems experienced when using traditional RNNs [45], [46]. The quantity of neurons in the input layer is dictated by the feature space size, as depicted in Fig. 3. Similarly, the output space corresponds to the number of neurons in the output layer. The memory cells in the hidden layer(s) cover the primary functions of GRU networks. Variations and preservation of cell status depend on two gates in the cell: the update gate r_t, and an input gate z_t [45], [47], [48]. The following equations (i.e. (1) and (2)) define the GRU architecture as a whole.

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right) \quad (1)$$

$$r_t = \sigma\left(W_{r.}\, [h_{t-1}, x_t]\right) \quad (2)$$

$$\tilde{h}_t = \tanh\left(W_h \cdot [r_t * h_{t-1}, x_t]\right) \quad (3)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \bar{h}_t \quad (4)$$

where the input gate is represented by $_zt$ and the state vector for a time frame t is represented as $\mathrm{h_t}$. The activation function, which keeps the reset and update gates within the range of Zero (0) and One (1), is represented by σ , x_t is the input vector at a time and W_z, W_r, W_h are matrices of the model parameters. Before the non-linearities are applied, the architecture adds a set of bias vectors b_z, b_r, *and* b_h that can be trained [47], [49], [50].

The motivation for Bi-directional Long Short-Term Memory (BiLSTM) stems from RNN simultaneously processing information in both directions. To process input sequences in both the forward and backward directions, bidirectional RNNs use two distinct sets of hidden layers. As shown in Fig. 4, multiple time steps in succession represent a BiLSTM structure. Each hidden layer in a BiLSTM is linked to the same output layer. BiLSTM permits bidirectional information flow to mitigate RNN's limitation of using only the prior context of the input data sequence [51].

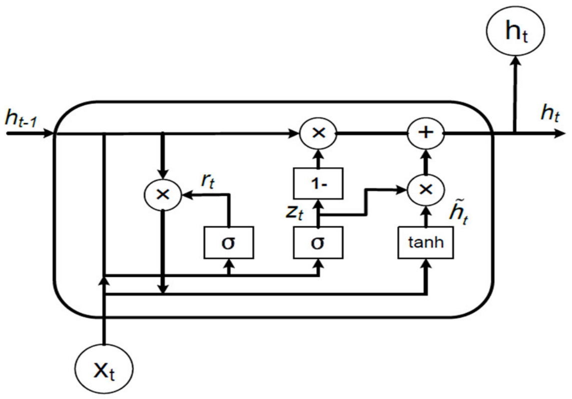


FIGURE 3. Gated recurrent unit structure [42], [43].

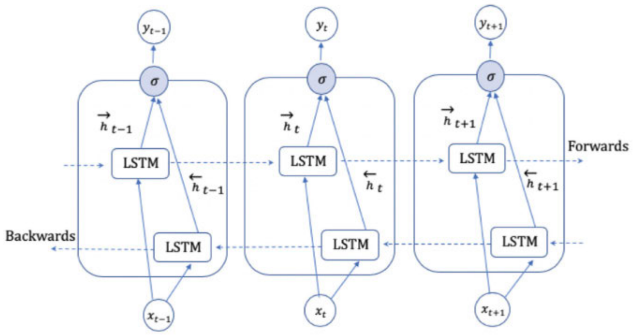


FIGURE 4. BiLSTM architecture with three layers [48].

Its updating mechanism is usually defined by (5) through (7) where $\vec{h}(t)$ is the forward hidden sequence layer, $\overleftarrow{h}(t)$ is the backward hidden sequence, and $y(t)$ represents the output. To arrive at an estimate, the network performs iterations on the forward layer from t = 1 to $\mathrm{t_f}$ and the backward hidden layer from t = $\mathrm{t_f}$ to 1.

$$\vec{h}(t) = H\left(W_{\vec{j}}X_t + V_{\vec{j}}h_{\vec{j}}(t-1) + b_{\vec{j}}\right) \quad (5)$$

$$\overleftarrow{h}(t) = H\left(W_{\vec{j}}X_t + V_{\vec{j}}h_{\vec{j}}(t-1) + b_{\bar{j}}\right) \quad (6)$$

$$y(t) = U_{\bar{j}}h_j(t) + U_{\bar{j}}h_j^{-}(t) + b_y \quad (7)$$

The final output, which is y, is computed as

$$\mathbf{y}(t) = \sigma_y(\vec{h}, \overleftarrow{h}) \quad (8)$$

A model's effectiveness is determined by calculating the confusion matrix, from which various performance metrics can be computed such as True Positive Rate (TPR), False Positive Rate (FPR), False Negative Rate (FNR), Accuracy/Classification Rate (CR), Precision/Positive Predictive

Value (PPV), Specificity/True Negative Rate (TNR), F1 Score, Training time and Testing time [49], [50].

D. RELATED WORKS

Talaei Khoei and Kaabouch [11] developed a novel outlier-based method for detecting zero-day cyberattacks. The primary objective was to create an intelligent IDS model proficient at sensing zero-day cyberattacks with high detection precision. Evaluation of the model using the CIC-IDS2017 and NSL-KDD dataset shows a detection accuracy of 90.01% for DoS (GoldenEye), 98.43%, for DoS (Hulk), 90.01%, for port scanning, and 99.67% for DDoS attacks. Sharafaldin et al. [38] proposed the Hierarchical Deep Learning System based on Big Data (BDHDLS). The BDHDLS employs behavioral and content features to comprehend network traffic characteristics and payload information. Each DL model in the BDHDLS focuses on learning the specific data distribution in a single cluster. This strategy boosts the detection rate of intrusive attacks compared to prior approaches that utilized a single learning model. The model was trained on the Information Centre of Excellence for Tech Innovation (ISCX) Intrusion Detection Assessment dataset (ISCXIDS2012) and achieved an accuracy rate of 99.5%.

Lee et al. [39] developed a paradigm for adaptive ensemble learning. The authors developed a multi-tree algorithm by modifying the amount of training data and establishing numerous decision trees. They selected multiple basic classifiers such as Decision Trees, Random Forests, K-Nearest Neighbors (KNN), and Deep Neural Networks (DNN) to develop an ensemble adaptive voting algorithm to enhance the overall detection effect. The authors used the NSL-KDD to validate their proposed approach. From their findings, the multi-tree algorithm accuracy was 84.2%, while the adaptive voting algorithm had an accuracy of 85.2%. In [40], the researchers proposed a model for network intrusion detection based on the Convolutional Neural Network (CNN) method. The model automatically extracts the effective characteristics of intrusion samples, allowing for their proper classification. It combines convolution and pooling operations to extract feature correlations among the data more effectively. Evaluation with the KDD99 datasets indicated that the proposed model achieved an accuracy of 99.23%. Ring et al. [41] proposed a CNN–IDS-based network intrusion detection model. The model used CNN to automatically extract dimensionally reduced data features. The authors utilized a typical KDD-CUP99 dataset to evaluate the CNN model. From the findings, the CNN model attained an accuracy of 94%, a Detection Rate of 93%, and a False Alarm Rate of 0.5%.

In [42], the authors developed a new IDS that integrates various classifier approaches based on decision tree and rule-based algorithms, such as the Reduced Error Pruning (REP) tree, Repeated Incremental Pruning (RIP) algorithm, and Forest by Penalizing Attributes (Forest PA) algorithm, to identify suspicious activity. Benign network traffic was classified using the first and second methods, which use the dataset's features as inputs. The third classifier considers aspects of the initial dataset and the results of the first and second classifiers as inputs. With a training time of 195.5 seconds and a test duration of 2.27 seconds, the proposed IDS model attained an accuracy of 96.67%, 94.48% detection rate and 1.15% false positive on the CICIDS2017 dataset. Chattopadhyay et al. [43] proposed an IDS that uses Core Vector Machines (CVM), a data mining-based classifier. Compared to Support Vector Machine (SVM) and ensemble classifiers, it has a lower false positive rate and a lower computational overhead. The classifier was trained and tested on the KDDCup'99 dataset. Evaluation results show a detection rate of 99% and a false positive rate of 27%. Shen et al. [45] presented a Genetic Algorithm (GA) and SVM-based alarm intrusion detection algorithm for use in a human-controlled IDS. Using GA's crossover and mutation probabilities as a starting point, the authors improved both the efficiency of the population search and the ability of individuals to share information. The algorithm's convergence time and SVM training speed were both enhanced.

Xu et al. [46] proposed an enhanced IDS based on a hybrid of PCA and Gaussian Naive Bayes for intrusion detection. The PCA component reduces data pollution by weighing the first few principal feature vectors, thus improving data dimensionality. The Gaussian Naive Bayes classifier detected intrusion behaviours. The model evaluation shows a detection reduction time of 60% and an increase in the detection rate of 91.06%. Ravanelli et al. [47] proposed an IDS using an enhanced Genetic K-Means (IGKM) algorithm. This article compares an intrusion detection system using the k-means++ algorithm to one using the IGKM algorithm using a smaller portion of the KDD99 dataset with 1,000 occurrences. The k-means++ approach achieves 53.27% accuracy, while IGKM achieves 72.91%. Lu et al. [48] proposed LA-GRU as a new Intrusion Detection Model (IDM) based on unbalanced learning and Gated Recurrent Unit (GRU) networks. In the suggested approach, a modified Local Adaptive Synthetic Minority Oversampling Technique (LA-SMOTE) algorithm handles imbalanced traffic. The GRU network based on deep learning theory detects traffic anomalies. Evaluation of the NS-KDD dataset shows that the proposed model got an accuracy of 99.04%, a Detection Rate of 98.92%, and an FPR of 0.13%.

Zhao et al. [12] developed the RNN-IDS model that is adept at intrusion detection modelling and has excellent accuracy in binary and multi-class classification. Compared to conventional classification techniques such as J48, Naive Bayes, and random forest, the model achieves a higher accuracy rate and detection rate of 99.81% and 97.09%, respectively. Lynn et al. [49] proposed a new ensemble classifier for IDS. The classifier was developed using the well-known Random Forest (RF) and Average One-Dependence Estimator (AODE). The AODE handled the issue of attribute reliance in the Naive Bayes classifier, while the RF increased the precision rate. The performance of the proposed ensemble classifier (RF+AODE) on the Kyoto data shows an accuracy of 90.51% and a false alarm rate of 0.14 over AODE, Nave Bayes, and RF algorithms.

Khan et al. [36] proposed to improve Unmanned Aerial Vehicle (UAV) performance using a blockchain-based decentralized ML architecture. The UAV's method for intelligent decision-making can enhance data storage and integrity. The authors analyzed the system using CIDS to demonstrate the viability and efficacy of applying blockchain-based decentralized ML in UAVs and similar applications. They utilized the KDD99 threat dataset, dividing 10,000 rows over the active nodes of the blockchain network to cover various attack techniques, with an accuracy of 99.6%. Liu et al. ([54] suggested a method that offloads model training to edge devices as a cooperative IDS. Distributed and federated architecture reduces central server resource use and ensures security and privacy. Blockchain stores and shares the training data to secure the aggregate model. KDDCup99 dataset accuracy was 86.7%. The implementation focused on the public Ethereum blockchain consensus mechanism. In [55], the authors proposed the Deep Blockchain Framework (DBF), which offers CIDS and blockchain with smart contracts in the cloud. The intrusion detection method was measured using the UNSW-NB15 dataset and a BiLSTM DL algorithm to handle time-series network data. Using the Ethereum library, intelligent contract mechanisms and privacy-based blockchain protect distributed IDS. The model's accuracy increased from 97.26% to 99.41%.

In [56], the authors proposed a model that utilized DL and Blockchain to preserve smart power network datasets and detect assaults. The authors included privacy at two levels; the first level uses enhanced Proof of Work (ePoW) to verify data integrity; the second uses a variational autoencoder to encode and alter data. Two degrees of privacy yielded higher performance than prior methods and avoided data poisoning and inference assaults on smart power network datasets. The model was trained on the UNSW-NB15 dataset and achieved a classification Rate of 99.80% and FAR of 0.01. Liang et al. [57] developed a deep learning IDS that utilized blockchain for a multi-agent system. The study designed, implemented and tested an IDS employing a multi-agent system, blockchain, and DL techniques. The system module comprises data collection, management, analysis, and reaction. The results reveal a validation accuracy of 98.27%, a testing accuracy of 83.17%, a precision of 83.50%, a recall of 84.14%, and F1 Score of 83.94% on the NSL-KDD dataset.

To the best of the authors' knowledge, no previous study has explored the use of a secure and private intrusion detection scheme with private blockchain technology within a federated cloud infrastructure. Thus, it has become highly critical to address this gap, given the benefits of cloud federation for more effective utilization of computing resources.

III. DESIGN AND IMPLEMENTATION

The proposed architecture and its various components are presented and discussed in this section. The architecture in

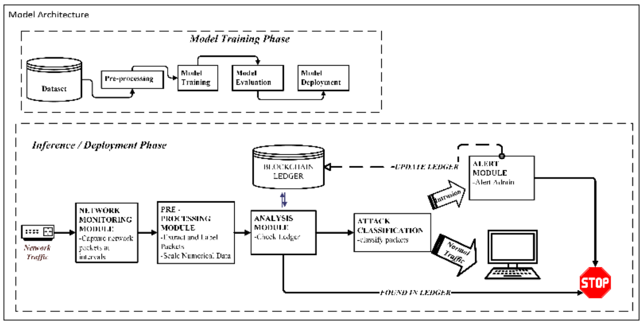

Model Architecture
Model Training Phase
Dataset
Pre-processing
Model Training
Model Evaluation
Model Deployment
Inference / Deployment Phase
BLOCKCHAIN LEDGER
-UPDATE LEDGER
ALERT MODULE -Alert Admin
Intrusion
Network Traffic
NETWORK MONITORING MODULE -Capture network packets at intervals
PRE - PROCESSING MODULE -Extract and Label Packets -Scale Numerical Data
ANALYSIS MODULE -Check Ledger
ATTACK CLASSIFICATION -classify packets
Normal Traffic
STOP
FOUND IN LEDGER

FIGURE 5. Architecture of the proposed SecFedIDM-V1.

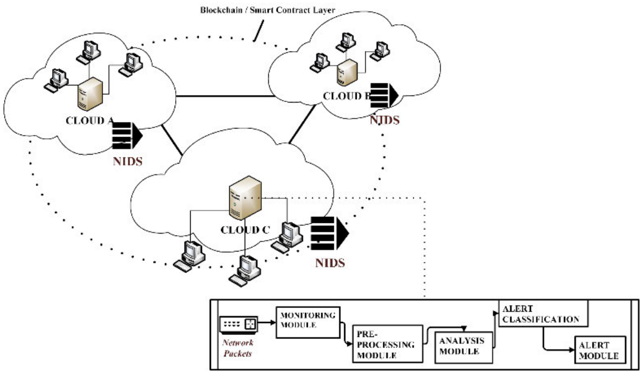


FIGURE 6. Architecture of the proposed SecFedIDM-V1 within a federated cloud infrastructure.

Fig. 5 is that of the proposed model deployed in a single cloud platform, while Fig. 6 shows the SecFedIDM-V1 deployed in a Cloud Federation, with each cloud region having the model deployed as an NIDS (as indicated in the diagram). The Blockchain technology allows all the region logs to be automatically replicated in the others securely.

A. DATASET CURATION AND PRE-PROCESSING

As earlier mentioned, we curated the dataset for this work from the Coburg Intrusion Detection Data Sets (CIDDS) [41], [42]. Notably, CIDDS was originally generated from an OpenStack-based cloud platform, which means that models developed based on it will fit into similar cloud platforms (such as the FEDGEN Testbed [54], [55]). Table 1 presents the attributes of CIDDS. The steps for data curation and pre-processing in this study are hereafter presented.

Step 1: Import the dataset

The CIDDS dataset comes in four (4) Comma Separated Values (CSV) files. First, a variable holds the location of the files, and another is created as an empty list that will hold the imported data files. An iterative procedure is applied to the paths, the files are appended and then saved into the data files list.

Step 2: Checking for missing values and inconsistent data in the dataset

All rows that contain missing values, Not a Number (NaN) cells, and inconsistent data are dropped from the dataset.

TABLE 1. Attributes of the cidds dataset.

SN	Attributes	Description
1.	Src IP	IP Address of Source
2.	Src Port	Port used on the Source
3.	Dest IP	IP Address of Destination
4.	Dest Port	Port used on the Destination
5.	Proto	ICMP, TCP, or UDP are the transport protocol
6.	Date first seen	Start time flow first seen
7.	Duration	Time taken for the flow to complete
8.	Bytes	Total bytes transmitted
9.	Packets	Total packets transmitted
10.	Flags	OR chain of all TCP Flags
11.	Class	Label for classes
12.	AttackType	Attack type which could be port Scan, DoS or brute Force
13.	AttackID	Unique Attack ID for each attack type
14.	Attack Description	Offers extra information about the set attack parameters

TABLE 2. Selected features for network training.

SN	Feature	Description	Data type
1.	Src Port	Port used on the Source	Float
2.	Dest Port	Port used on the Destination	Float
3.	Protocol	ICMP, TCP, or UDP are the transport protocol	Float
4.	Duration	Duration of the flow	Float
5.	Bytes	Total transmitted bytes	Float
6.	Packets	Total transmitted packets	Float
7.	AttackType	port Scan, dos, brute Force & normal	Object/String

Step 3: Encoding categorical data

Machine learning is more adept at learning from numerical than categorical data. Encoding categorical data is turning categorical data into an integer format so that the data containing the converted categorical values can be fed to models to improve their predictions. The AttackType and Proto columns of the dataset were encoded using the one-hot encoding technique as used in [48].

Step 4: Features Selection

The CIDDS dataset has a total of 14 features. The Pearson Correlation Coefficient (PCC) was applied to the dataset as a feature selection method to remove the highly correlated columns. Afterwards, Seven (7) of the 14 features were selected as shown in Table 2.

Step 5: Scaling/Normalization of Numerical Data

Scaling/normalization helps to make the flow of gradient descent seamless and expedites algorithms' convergence at a minimal cost function. In this study, minMaxScaler, expressed in (9), was employed to normalise the dataset. x represents a feature, x_i represents the present value of x and x_i^{new} is the normalized value.

$$x_i^{new} = \frac{x_i - min(x)}{max\ (x) - min(x)} \tag{9}$$

TABLE 3. Distribution for training, test, and validation sets.

	60:20:20	70:15:15	80:10:10
Training Set	13,269,720	15,481,351	17,692,983
Testing Set	4,423,241	3,317,431	2,211,621
Validation Set	4,423,241	3,317,420	2,211,598
TOTAL	22,116,202	22,116,202	22,116,202

Step 6: Train, Test and Validation subsets

After pre-processing, the dataset was left with 22,116,202 records. The records were split into three categories for training, testing and validation respectively. The categories are i) 60%:20%:20%, ii) 70%:15%:15% and iii) 80%:10%:10% as shown in Table 3.

B. DESIGN OF THE DEEP LEARNING MODELS

Definition 1: Learning in ML or DL can be supervised, unsupervised and semi–supervised depending on the dataset's structure. CIDDS is a labelled dataset that is suitable for supervised learning. The CIDDS dataset is split into the training, testing, and validation subsets, $\{(x_j, C_j)\}$ with $j \in \{1, \ldots, N\}$ where x_j are samples in each of the subsets, and N is the total samples with an equivalent number of class labels $c_j \in \{1, \ldots, m\}$ where the number of classes is m. In this work, N = 22,116,202 and m = 5 (i.e., normal, port Scan, DOS, Ping Scan and brute Force).

Definition 2: BiLSTM and GRU networks operate optimally in sequential learning jobs while avoiding gradient vanishing and explosion challenges that affect RNNs when learning long-term dependencies [45], [58]. The BiLSTM and GRU architectures were compared in this study to determine the appropriate one for implementing the traffic sequence classification task in the proposed SecFedIDM-V1. Given x as a timestep input with a 2-dimensional feature vector, g as the activation function, Was the weight, w_o as the bias, the output $\hat{y}$, of the model can be expressed as represented in (10).

$$\hat{y} = g(w_o + X^T W) \tag{10}$$

$$\text{where X} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_f \end{bmatrix} \text{and W} = \begin{bmatrix} w_1 \\ w_2 \\ . \\ . \\ . \\ w_f \end{bmatrix}$$

The GRU and BiLSTM models were implemented using the same configuration. Both utilize a dropout layer with 0.1 dropouts and a dense layer in the hidden layer, as shown in Fig. 7 and Fig. 8. The models are multi-class as the classification outputs are: *'normal,' 'portScan,' 'dos,' 'pingScan,'* or *'bruteForce,'*and the *SoftMax* activation function is suitable for such. The detailed model configuration used in the model training phase is given in Table 4. The Rectified Linear Unit

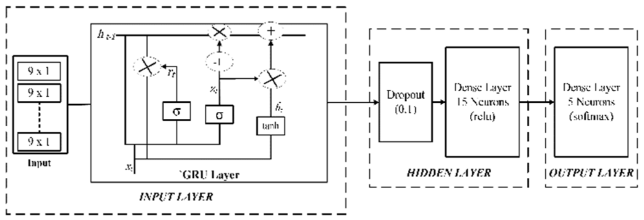


FIGURE 7. Architecture of the GRU network.

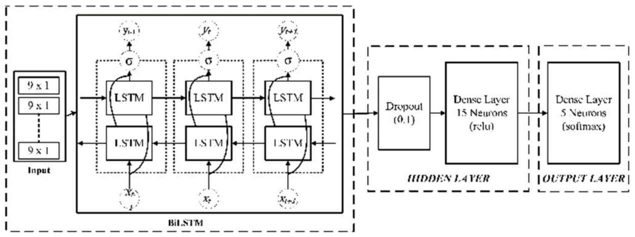


FIGURE 8. Architecture of BiLSTM network.

TABLE 4. Model training configuration.

SN	Quantity	Configuration
1	Input	*ReLU* activation function
2	Dropout	0.1
3	Hidden Layer	*ReLU* activation function
4	Output	*SoftMax* activation function
5	Optimizer	Adam
6	Loss	Categorical Cross Entropy
7	Metrics	AUC, Precision, TP, TN, FN, FP
8	Batch Size	256
9	Epochs	100

(ReLU) activation function, as expressed in (11), was used at the input and hidden layers, while the *SoftMax* activation function, expressed in (12), was used at the output.

$$g(z) = \max(0, z) \tag{11}$$

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e_j^z} \tag{12}$$

where g is the activation function while z is the sum of weighted input to the function. The *SoftMax* function takes in z as its input, which comprises z_0 to z_K whereas z_i has an upper limit of 1, a lower limit of 0 and the number of classes in the classifier is K. The summation of the outputs using *Softmax* results in 1.

C. NETWORK MONITORING AND PACKET CAPTURE

The SecFedIDM-V1 is embedded in a cloud-native web application and deployed on a federated cloud testbed to monitor the network and classify it as either normal or attack. The FEDGEN Testbed [54], which is the experimental federated cloud platform in this work runs OpenStack as the cloud operating system. The incoming traffic is captured and sent to a well-trained sequential network for proper classification after pre-processing. Fig. 9 shows the location of the Packet Sniffer, which is connected to Neutron (OpenStack network service) to monitor and capture the network traffic, which is then sent to the SecFedIDM-V1 for classification of the traffic.

TABLE 5. Configured entities.

SN	Entity/Organization	Peers	Role	Organisation MSP ID
1	FRA	Peer0	Committer	AlphaMSP
2	FRA	Peer1	Endorser	AlphaMSP
3	FRB	Peer0	Committer	BravoMSP
4	FRB	Peer1	Endorser	BravoMSP
5	FRC	Peer0	Committer	CharlieMSP
6	FRC	Peer1	Endorser	CharlieMSP

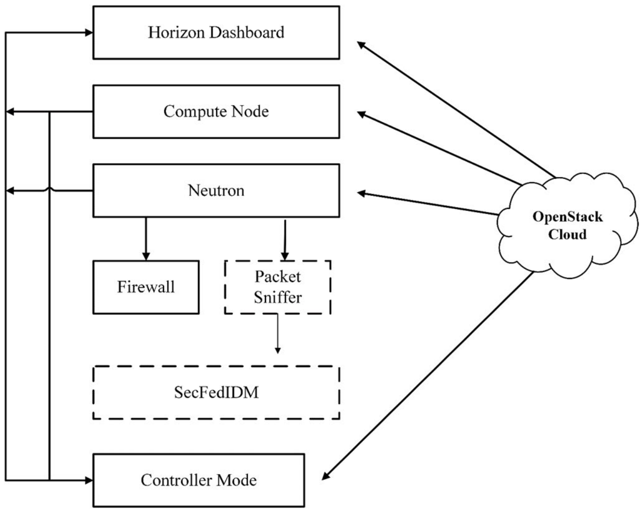


FIGURE 9. Packet sniffer and SecFedIDM-V1 location in OpenStack.

D. BLOCKCHAIN COMPONENT OF THE SECFEDIDM-V1 ARCHITECTURE

Hyperledger fabric was used as a security layer to ensure that records/logs were safe and secured. The first step is defining the business network, which entails identifying the participants, their functions, and their stake in the enterprise. Users, their responsibilities, and affiliations or departments must all be created. The hypothetical organizations that were configured on the FEDGEN Testbed are:

i. FEDGEN Region Alpha (FRA);
ii. FEDGEN Region Bravo (FRB); and
iii. FEDGEN Region Charlie (FRC).

The configuration of the various entities is shown in Table 5.

TABLE 6. software packages required to implement the hyperledger fabric.

SN	Software Package	Version	Function
1	cURL	7.81.0	Client URL (cURL), is a PowerShell tool that developers use to transfer data to and from a server.
2	NPM	8.5.1	It is a package manager for the JavaScript programming language.
3	Docker	20.10.17	It is an open-source containerization platform which allows developers to package applications into containers.
4	Docker Compose	1.29.2	Docker Compose was created to aid in the definition and sharing of multi-container applications.
5	Golang	1.18.1	Golang Programming Language is used in writing the chaincode.
6	Node	12.22.9	A cross-platform and open-source JavaScript runtime environment.

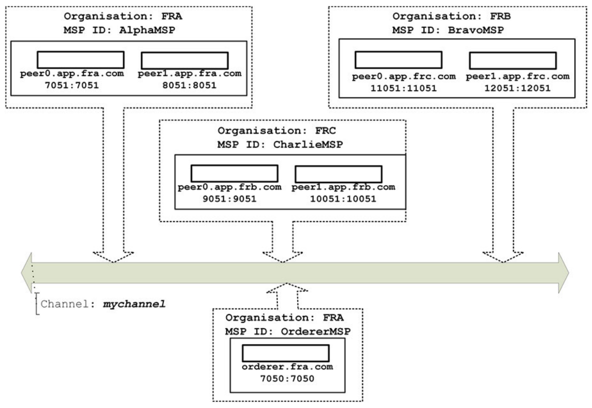


FIGURE 10. Blockchain network topology.

The table contains the organisation's role and the Membership Service Providers Identity (MSP ID). The MSP ID is an arbitrary value that is used in identifying each organisation.

Designing a network topology is a significant step in implementing blockchain in the enterprise. The network comprises six peer nodes representing three different organizations (nodes), as presented in Table 5 and an Orderer node. The network also depicts a channel that peers will use to validate and execute transactions. The Orderer node is responsible for guaranteeing transaction order and delivery. A channel is created using the Orderer service, allowing for a common communication infrastructure for peer and client nodes, The Orderer node is part of the FRA organization (orderer.fra.com) as shown in Fig. 10.

In Hyperledger Fabric terminology, smart contracts are called chain codes. It has business functions invoked as part of transactions to query and update the ledger's state. The chain code is termed '*transcc.*' It consists of necessary business logic methods invoked to carry out the recording of attacks in a ledger. Chain codes are installed on each peer and run as an isolated docker container. Every chaincode must implement two key methods from the Hyperledger Fabric interface. The *init*() method is invoked when a client application instantiates, the method handles the initialization of the chaincode with predefined values while the invoke method handles the internal method allocation to handle client application requests.

For the deployment of the blockchain, the topology was set up using Fabric-based tools and commands. The steps for realising the deployment are hereafter presented.

Step 1: Generating crypto keys and certificates

The first step is to generate keys and certificates for every organization and its associated entities like peers and orderers so that they can be used to sign/verify as they communicate over the network. The *'cryptogen'* tool is used to generate the necessary certificates. The *'cryptogen'* tool reads the *'crypto-config.yaml'* file to generate the certificates. The *'crypto-config.yaml'* file depicts the network topology in a declarative way.

Step 2: Generating transaction configuration

This involves the creation of transaction configuration artifacts. These are genesis block and channel transaction configurations. The *'configtxgen'* tool is used to generate the artifacts. The said tool reads the '*configtx.yaml*' file. The genesis block is the first block in the blockchain, it is the configuration block that initializes the ordering service. The channel transaction configuration is used to create the channel to be used by the network.

Step 3: Starting the network

The network is started by spinning up the nodes. Docker images provided by Fabric runtime are used to start the nodes. Docker provides a consistent containerised environment to work with different platforms like MacOS, Linux and Windows.

Step 4: Creating a channel

The channel creation process reads the previously generated channel transaction configuration artifact to create the channel. The *channel creates* command reads the previously configured artifact *'channel.tx'* and creates the channel with the channel name *'mychannel'*. The channel name can however be any string of choice.

Step 5: Joining peers to the channel

Once the channel is created, the peer nodes should formally join the channel to participate in the transactions on that channel. The *peer join* command is supplied with the channel block file with the '-b' option to specify the channel block file. Since there are 6 peer nodes, this command must run six times and the value of the CORE_PEER_ADDRESS environment changed to point to the correct peer node address each time it is run.

Step 6: Installing the chaincode

This step involves the installation of the chaincode file on every peer node. The peer node endorses and verifies transactions invoked using the chaincode. The chaincode is installed in the file system of the peer node. Peer node address can be specified by setting the CORE_PEER_ADDRESS

TABLE 7. Precision, recall and F1 scores of BiLSTM and GRU.

	BiLSTM			GRU		
	60:20:20	**70:15:15**	**80:10:10**	**60:20:20**	**70:15:15**	**80:10:10**
Precision	0.98151	0.99616	***0.99624***	0.99451	0.98437	0.99263
Recall	0.99538	0.99904	***0.99906***	0.99439	0.98425	0.99250
F1 Score	0.98137	0.99608	***0.99614***	0.99451	0.98437	0.99263

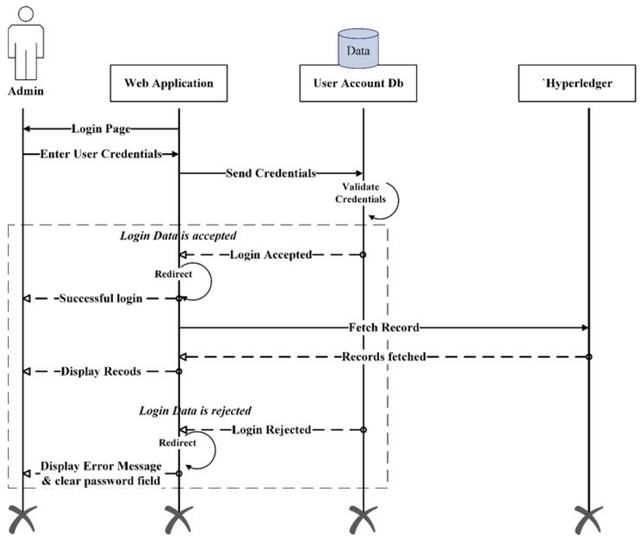


FIGURE 11. Sequence diagram for the cloud native web application.

environment variable. Since there are six peer nodes, the code runs this command six times with each time updating the environment variable.

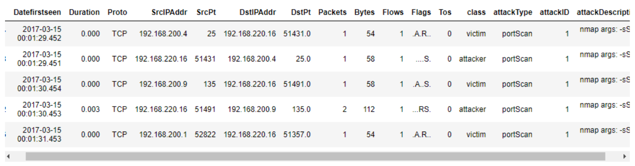
Step 7: Instantiating the chaincode

Now, that the chaincode has been installed on each peer, the 7th step involves the instantiation of the chaincode. Instantiating the chaincode means initializing the chaincode by invoking its *init*() method. Table 6 shows the various software packages used to implement the Hyperledger fabric.

Furthermore, a cloud-native web application was built around the SecFedIDM-V1. Fig. 11 shows the sequence diagram of the web application. The deployment involves the creation of a docker image from a docker file, which contains the list of dependencies i.e., the software packages listed in Table 6. The docker file also contains a list of instructions that need to be executed in a sequence for the development environment to be created. The containerised docker image was then deployed to our OpenStack-based federated cloud testbed (i.e. FEDGEN Testbed) as a proof of concept. The image can also be deployed to similar federated cloud platforms or reconfigured based on the peculiarities of any target platform. With the running application, an administrator can carry out network monitoring and packet capture to provide notifications on intrusions.

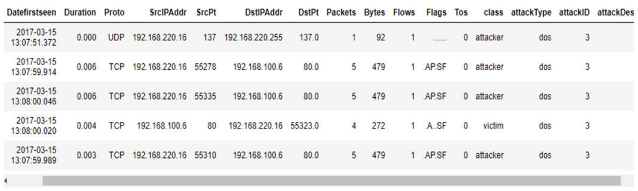
IV. RESULTS AND DISCUSSION

In this section, the implementation as experimental results for the different components of SecFedIDM-V1 are presented and discussed. As earlier established, the processed dataset



Datefirstseen	Duration	Proto	SrcIPAddr	SrcPt	DstIPAddr	DstPt	Packets	Bytes	Flows	Flags	Tos	class	attackType	attackID	attackDescripti
2017-03-15 00:01:29.452	0.000	TCP	192.168.200.4	25	192.168.220.16	51431.0	1	54	1	.A.R..	0	victim	portScan	1	nmap args: -sS
2017-03-15 00:01:29.451	0.000	TCP	192.168.220.16	51431	192.168.200.4	25.0	1	58	1	....S.	0	attacker	portScan	1	nmap args: -sS
2017-03-15 00:01:30.454	0.000	TCP	192.168.200.9	135	192.168.220.16	51491.0	1	58	1	.A..S.	0	victim	portScan	1	nmap args: -sS
2017-03-15 00:01:30.453	0.003	TCP	192.168.220.16	51491	192.168.200.9	135.0	2	112	1	...RS.	0	attacker	portScan	1	nmap args: -sS
2017-03-15 00:01:31.453	0.000	TCP	192.168.200.1	52822	192.168.220.16	51357.0	1	54	1	.A.R..	0	victim	portScan	1	nmap args: -sS

FIGURE 12. Attribute values of the PortScan class in CIDDS.



Datefirstseen	Duration	Proto	SrcIPAddr	SrcPt	DstIPAddr	DstPt	Packets	Bytes	Flows	Flags	Tos	class	attackType	attackID	attackDes
2017-03-15 13:07:51.372	0.000	UDP	192.168.220.16	137	192.168.220.255	137.0	1	92	1	......	0	attacker	dos	3	
2017-03-15 13:07:59.914	0.006	TCP	192.168.220.16	55278	192.168.100.6	80.0	5	479	1	.AP.SF	0	attacker	dos	3	
2017-03-15 13:08:00.046	0.006	TCP	192.168.220.16	55335	192.168.100.6	80.0	5	479	1	.AP.SF	0	attacker	dos	3	
2017-03-15 13:08:00.020	0.004	TCP	192.168.100.6	80	192.168.220.16	55323.0	4	272	1	.A..SF	0	victim	dos	3	
2017-03-15 13:07:59.989	0.003	TCP	192.168.220.16	55310	192.168.100.6	80.0	5	479	1	.AP.SF	0	attacker	dos	3	

FIGURE 13. Attribute Values of the DoS Class in CIDDS.

TABLE 8. FPR, FNR and TPR of BiLSTM and GRU.

	BiLSTM			GRU		
	60:20:20	**70:15:15**	**80:10:10**	**60:20:20**	**70:15:15**	**80:10:10**
FPR	0.00462	0.00096	***0.00094***	0.00137	0.00391	0.00184
FNR	0.01877	0.00399	***0.00395***	0.00572	0.01586	0.00762
TPR	0.98123	0.99601	***0.99605***	0.99428	0.98414	0.99238

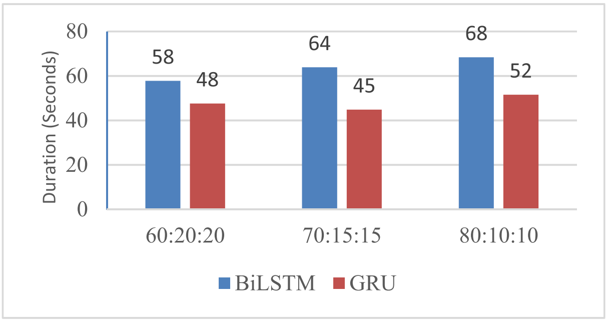
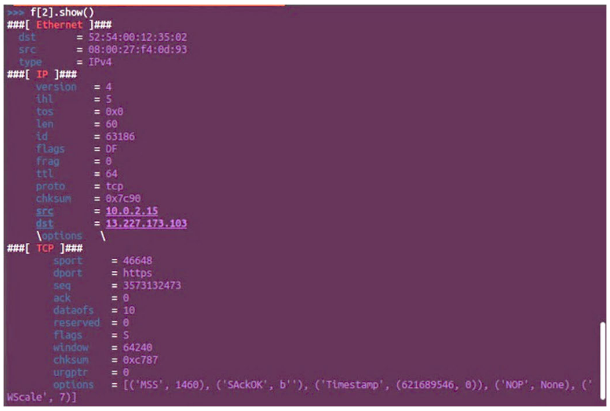


FIGURE 14. Training time for BiLSTM and GRU models.

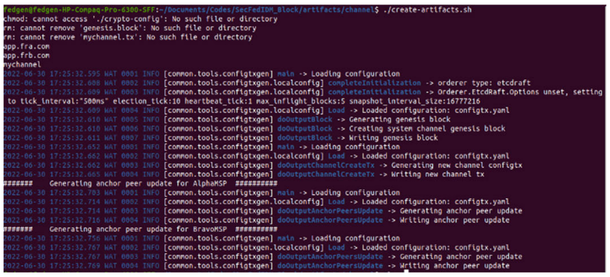
was split into training, testing, and validation ratios. Fig. 12 shows the output for the PortScan class in the dataset, and Fig. 13 shows the output for the DoS class. As expected, the values of corresponding attributes in the two classes are unique. This is also the case with other classes in the dataset based on the feature selection procedure that was implemented at the pre-processing stage. The model training and testing were done sequentially to avoid overwhelming the system resources. Table 7 shows the BiLSTM and GRU classification's Precision, Recall, and F1 scores. It can be observed that the BiLSTM network outperformed GRU by achieving a precision of 0.99624, a recall rate of 0.99906, and an F1 of 0.99614 for the data partition with a ratio of 80:10:10. Furthermore, Table 7 shows that the 80:10:10 partition ratio for BiLSTM gave better performance than the other two partition ratios that were experimented. For instance, the 70:15:15 ratio achieved a precision of 0.99616, a recall rate of 0.99904 and an F1 score of 0.99608 while the 60:20:20 ratio achieved a precision of 0.98151, a recall rate of 0.99538 and an F1 score of 0.98137.

Additionally, the models were evaluated based on FPR, FNR and TPR. An ideal model should have high TPR, low



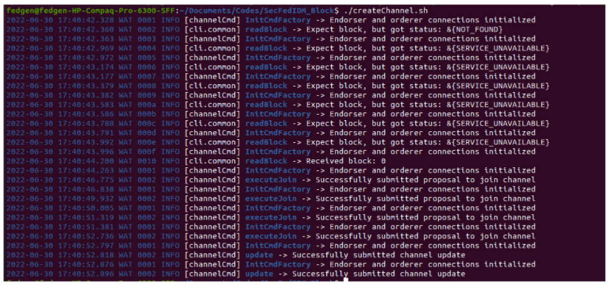
```
>>> f[2].show()
###[ Ethernet ]###
  dst       = 52:54:00:12:35:02
  src       = 08:00:27:f4:0d:93
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 60
     id        = 63186
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x7c90
     src       = 10.0.2.15
     dst       = 13.227.173.103
     \options   \
###[ TCP ]###
        sport     = 46648
        dport     = https
        seq       = 3573132473
        ack       = 0
        dataofs   = 10
        reserved  = 0
        flags     = S
        window    = 64240
        chksum    = 0xc787
        urgptr    = 0
        options   = [('MSS', 1460), ('SAckOK', b''), ('Timestamp', (621689546, 0)), ('NOP', None), ('
WScale', 7)]
```

FIGURE 15. Sample of a captured packet by scapy.



```
fedgen@fedgen-HP-Compaq-Pro-6300-SFF:~/Documents/Codes/SecFedIDM_Block/artifacts/channel$ ./create-artifacts.sh
chmod: cannot access './crypto-config': No such file or directory
rm: cannot remove 'genesis.block': No such file or directory
rm: cannot remove 'mychannel.tx': No such file or directory
app.fra.com
app.frb.com
mychannel
2022-06-30 17:25:32.595 WAT 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2022-06-30 17:25:32.608 WAT 0002 INFO [common.tools.configtxgen.localconfig] completeInitialization -> orderer type: etcdraft
2022-06-30 17:25:32.609 WAT 0003 INFO [common.tools.configtxgen.localconfig] completeInitialization -> Orderer.EtcdRaft.Options unset, setting
 to tick_interval:"500ms" election_tick:10 heartbeat_tick:1 max_inflight_blocks:5 snapshot_interval_size:16777216
2022-06-30 17:25:32.609 WAT 0004 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: configtx.yaml
2022-06-30 17:25:32.610 WAT 0005 INFO [common.tools.configtxgen] doOutputBlock -> Generating genesis block
2022-06-30 17:25:32.610 WAT 0006 INFO [common.tools.configtxgen] doOutputBlock -> Creating system channel genesis block
2022-06-30 17:25:32.611 WAT 0007 INFO [common.tools.configtxgen] doOutputBlock -> Writing genesis block
2022-06-30 17:25:32.652 WAT 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2022-06-30 17:25:32.662 WAT 0002 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: configtx.yaml
2022-06-30 17:25:32.662 WAT 0003 INFO [common.tools.configtxgen] doOutputChannelCreateTx -> Generating new channel configtx
2022-06-30 17:25:32.665 WAT 0004 INFO [common.tools.configtxgen] doOutputChannelCreateTx -> Writing new channel tx
#######    Generating anchor peer update for AlphaMSP  ##########
2022-06-30 17:25:32.703 WAT 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2022-06-30 17:25:32.714 WAT 0002 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: configtx.yaml
2022-06-30 17:25:32.714 WAT 0003 INFO [common.tools.configtxgen] doOutputAnchorPeersUpdate -> Generating anchor peer update
2022-06-30 17:25:32.716 WAT 0004 INFO [common.tools.configtxgen] doOutputAnchorPeersUpdate -> Writing anchor peer update
#######    Generating anchor peer update for BravoMSP  ##########
2022-06-30 17:25:32.756 WAT 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2022-06-30 17:25:32.767 WAT 0002 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: configtx.yaml
2022-06-30 17:25:32.767 WAT 0003 INFO [common.tools.configtxgen] doOutputAnchorPeersUpdate -> Generating anchor peer update
2022-06-30 17:25:32.769 WAT 0004 INFO [common.tools.configtxgen] doOutputAnchorPeersUpdate -> Writing anchor peer update
```

FIGURE 16. Output of create-artifacts command.



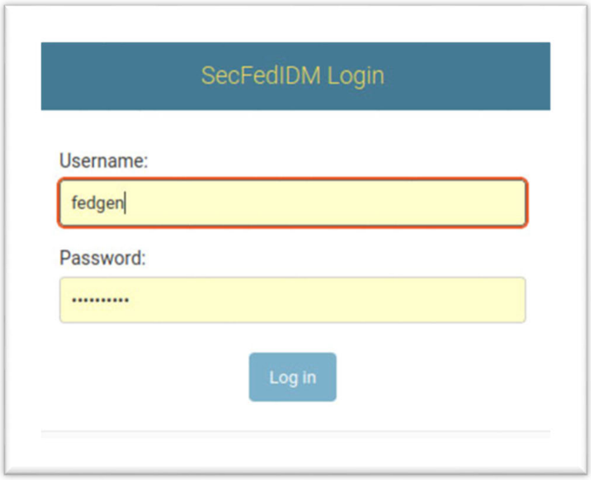
```
fedgen@fedgen-HP-Compaq-Pro-6300-SFF:~/Documents/Codes/SecFedIDM_Block$ ./createChannel.sh
2022-06-30 17:40:42.328 WAT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-06-30 17:40:42.360 WAT 0002 INFO [cli.common] readBlock -> Expect block, but got status: &{NOT_FOUND}
2022-06-30 17:40:42.363 WAT 0003 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-06-30 17:40:42.969 WAT 0004 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-06-30 17:40:42.972 WAT 0005 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-06-30 17:40:43.174 WAT 0006 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-06-30 17:40:43.177 WAT 0007 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-06-30 17:40:43.379 WAT 0008 INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-06-30 17:40:43.382 WAT 0009 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-06-30 17:40:43.583 WAT 000a INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-06-30 17:40:43.586 WAT 000b INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-06-30 17:40:43.788 WAT 000c INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-06-30 17:40:43.791 WAT 000d INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-06-30 17:40:43.992 WAT 000e INFO [cli.common] readBlock -> Expect block, but got status: &{SERVICE_UNAVAILABLE}
2022-06-30 17:40:43.996 WAT 000f INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-06-30 17:40:44.200 WAT 0010 INFO [cli.common] readBlock -> Received block: 0
2022-06-30 17:40:44.263 WAT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-06-30 17:40:46.775 WAT 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
2022-06-30 17:40:46.838 WAT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-06-30 17:40:49.932 WAT 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
2022-06-30 17:40:50.005 WAT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-06-30 17:40:51.319 WAT 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
2022-06-30 17:40:51.381 WAT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-06-30 17:40:52.736 WAT 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
2022-06-30 17:40:52.797 WAT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-06-30 17:40:52.818 WAT 0002 INFO [channelCmd] update -> Successfully submitted channel update
2022-06-30 17:40:52.876 WAT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-06-30 17:40:52.896 WAT 0002 INFO [channelCmd] update -> Successfully submitted channel update
```

FIGURE 17. Chaincode installed on all peers across the three regions.

FPR, and low FNR. From the results depicted in Table 8, the BiLSTM performs better compared to GRU achieving an FPR of 0.00094, FNR of 0.00395, and TPR of 0.99605. Thus, the BiLSTM is selected for the traffic sequence classification task in the proposed SecFedIDM-V1.

Notably, BiLSTM may require longer training time (see Fig. 14), but due to its architecture, it characteristically can give better performance than GRU for tasks that capture complex dependencies in network traffic sequences. Since the SecFedIDM-V1 is targeted at federated cloud platforms with sufficient computational resources, the training time does not pose any disadvantage. Nevertheless, the BiLSTM result in this study also gives improved performance over the result obtained in a similar study [53], which utilized a cascade of Naïve Bayes and Random Forest on the CCIDD dataset, with an accuracy of 97% and FPR of 0.0021.

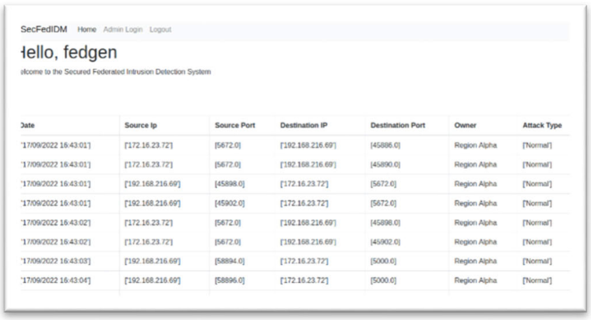
Furthermore, the Scapy library, an interactive packet manipulation library written in Python was used to monitor network traffic and capture packets at intervals of 90 seconds.



Username: fedgen

Password: ••••••••••

FIGURE 18. Login interface of the SecFedIDM-V1 cloud-native web application.



ɨello, fedgen

elcome to the Secured Federated Intrusion Detection System

Date	Source Ip	Source Port	Destination IP	Destination Port	Owner	Attack Type
'17/09/2022 16:43:01']	['172.16.23.72']	[5672.0]	['192.168.216.69']	[45886.0]	Region Alpha	['Normal']
'17/09/2022 16:43:01']	['172.16.23.72']	[5672.0]	['192.168.216.69']	[45890.0]	Region Alpha	['Normal']
'17/09/2022 16:43:01']	['192.168.216.69']	[45898.0]	['172.16.23.72']	[5672.0]	Region Alpha	['Normal']
'17/09/2022 16:43:01']	['192.168.216.69']	[45902.0]	['172.16.23.72']	[5672.0]	Region Alpha	['Normal']
'17/09/2022 16:43:02']	['172.16.23.72']	[5672.0]	['192.168.216.69']	[45898.0]	Region Alpha	['Normal']
'17/09/2022 16:43:02']	['172.16.23.72']	[5672.0]	['192.168.216.69']	[45902.0]	Region Alpha	['Normal']
'17/09/2022 16:43:03']	['192.168.216.69']	[58894.0]	['172.16.23.72']	[5000.0]	Region Alpha	['Normal']
'17/09/2022 16:43:04']	['192.168.216.69']	[58896.0]	['172.16.23.72']	[5000.0]	Region Alpha	['Normal']

FIGURE 19. The SecFedIDM-V1 dashboard.

As illustrated in the SecFedIDM-V1 architecture in Fig. 5, the captured traffic is pre-processed and classified by the trained BiLSTM network after the captured packets have been compared with the records in the blockchain ledger. Fig. 15 shows the snapshot of a captured IP/TCP packet with various details that the packet carries, ranging from source IP, destination IP, Time-to-Live (TTL), protocol type and other information.

Fig. 16 shows the successful creation of the blockchain network earlier presented in Fig. 10, the creation of the genesis block, and the transaction channel. In addition, Fig. 17 shows the output of the chaincode/smart contract installation on each organization's various peers as earlier presented in Table 5.

The login page of the Web Application that was implemented based on the SecFedIDM-V1 architecture using the Django framework is presented in Fig. 18. It provides authentication and authorization to prevent access to sensitive data on a federated cloud platform by unauthorised persons. The SecFedIDM-V1 dashboard showing network traffic features and the classified traffic type is shown in Fig. 19. In order to carry out deployment/production testing for the SecFedIDM-V1 based web application, traffic for the different attack classes were generated using network attack simulation tools such as *low orbit ion cannon, Nmap* and *John the ripper*.

V. CONCLUSION

In modern times, cloud federation has gained usage among Cloud Service Providers (CSP) as a means of sharing computational resources for better service delivery. However, with the increase in malicious activities and cyber-attacks, the need for secure transactions and communication between federated entities becomes critical. This study presents a BiLSTM-based IDS model for a federated cloud platform named SecFedIDM-V1. Firstly, we developed a cloud federation testbed using openstack. Secondly, we experimentally evolved an IDS model using CIDDS dataset with BiLSTM RNN giving the best performance metrics for intrusion traffic detection on an OpenStack-based federated cloud testbed. Thirdly, in order to enhance the security of the proposed architecture, we integrated a blockchain to serve as a secure datastore for intrusion signatures. In order to make the model usable for cloud system administrators, we developed a cloud native web application that incorporates the proposed BiLSTM IDS model. As a proof of concept, we illustrated how different classes of malicious traffic could be detected with high precision from the cloud application. The web app could be deployed on other experimental or production-federated cloud platforms to detect malicious intrusions. In the future, we hope to extend the architecture as well as the web application by covering novel and higher number of network attack classes.

REFERENCES

[1] M. R. M. Assis and L. F. Bittencourt, "A survey on cloud federation architectures: Identifying functional and non-functional properties," *J. Netw. Comput. Appl.*, vol. 72, pp. 51–71, Sep. 2016, doi: 10.1016/j.jnca.2016.06.014.

[2] O. Alkadi, N. Moustafa, and B. Turnbull, "A review of intrusion detection and blockchain applications in the cloud: Approaches, challenges and solutions," *IEEE Access*, vol. 8, pp. 104893–104917, 2020, doi: 10.1109/ACCESS.2020.2999715.

[3] D. Villegas, N. Bobroff, I. Rodero, J. Delgado, Y. Liu, A. Devarakonda, L. Fong, S. M. Sadjadi, and M. Parashar, "Cloud federation in a layered service model," *J. Comput. Syst. Sci.*, vol. 78, no. 5, pp. 1330–1344, Sep. 2012, doi: 10.1016/j.jcss.2011.12.017.

[4] M. J. Molo, J. A. Badejo, E. Adetiba, V. P. Nzanzu, E. Noma-Osaghae, V. Oguntosin, M. O. Baraka, C. Takenga, S. Suraju, and E. F. Adebiyi, "A review of evolutionary trends in cloud computing and applications to the healthcare ecosystem," *Appl. Comput. Intell. Soft Comput.*, vol. 2021, pp. 1–16, Sep. 2021, doi: 10.1155/2021/1843671.

[5] B. Rochwerger, C. Vázquez, D. Breitgand, D. Hadas, M. Villari, P. Massonet, E. Levy, A. Galis, I. M. Llorente, R. S. Montero, and Y. Wolfsthal, "An architecture for federated cloud computing," in *Cloud Computing—Principles and Paradigms*. Hoboken, NJ, USA: Wiley, 2011.

[6] T. Kurze, M. Klems, D. Bermbach, A. Lenk, S. Tai, and M. Kunze, "Cloud federation," in *Proc. 2nd Int. Conf. Cloud Comput., GRIDs, Virtualization*, 2011, pp. 32–38.

[7] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 416–464, 1st Quart., 2018, doi: 10.1109/COMST.2017.2771153.

[8] A. Jula, E. Sundararajan, and Z. Othman, "Cloud computing service composition: A systematic literature review," *Expert Syst. Appl.*, vol. 41, no. 8, pp. 3809–3824, Jun. 2014, doi: 10.1016/j.eswa.2013.12.017.

[9] P. Mishra, E. S. Pilli, V. Varadharajan, and U. Tupakula, "Intrusion detection techniques in cloud environment: A survey," *J. Netw. Comput. Appl.*, vol. 77, pp. 18–47, Jan. 2017, doi: 10.1016/j.jnca.2016.10.015.

[10] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, p. 20, Dec. 2019, doi: 10.1186/s42400-019-0038-7.

[11] T. Talaei Khoei and N. Kaabouch, "A comparative analysis of supervised and unsupervised models for detecting attacks on the intrusion detection systems," *Information*, vol. 14, no. 2, p. 103, Feb. 2023, doi: 10.3390/info14020103.

[12] J. Zhao, M. Chen, and Q. Luo, "Research of intrusion detection system based on neural networks," in *Proc. IEEE 3rd Int. Conf. Commun. Softw. Netw. (ICCSN)*, May 2011, pp. 174–178, doi: 10.1109/ICCSN.2011.6013688.

[13] S. A. R. Shah and B. Issac, "Performance comparison of intrusion detection systems and application of machine learning to snort system," *Future Gener. Comput. Syst.*, vol. 80, pp. 157–170, Mar. 2018, doi: 10.1016/j.future.2017.10.016.

[14] H. Hindy, R. Atkinson, C. Tachtatzis, J. N. Colin, E. Bayne, and X. Bellekens, "Utilising deep learning techniques for effective zero-day attack detection," *Electronics*, vol. 9, no. 10, pp. 1–16, 2020, doi: 10.3390/electronics9101684.

[15] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017, doi: 10.1109/ACCESS.2017.2762418.

[16] K. Dodo, L. T. Jung, M. A. Hashmani, and M. K. Cheong, "Empirical performance analysis of hyperledger LTS for small and medium enterprises," *Sensors*, vol. 22, no. 3, pp. 1–17, 2022, doi: 10.3390/s22030915.

[17] M. Hölbl, M. Kompara, A. Kamišalić, and L. N. Zlatolas, "A systematic review of the use of blockchain in healthcare," *Symmetry*, vol. 10, no. 10, p. 470, Oct. 2018, doi: 10.3390/sym10100470.

[18] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Blockchain for 5G and beyond networks: A state of the art survey," *J. Netw. Comput. Appl.*, vol. 166, Sep. 2020, Art. no. 102693, doi: 10.1016/j.jnca.2020.102693.

[19] N. A. Azeez, T. M. Bada, S. Misra, A. Adewumi, C. Van der Vyver, and R. Ahuja, "Intrusion detection and prevention systems: An updated review," in *Data Management, Analytics and Innovation* (Advances in Intelligent Systems and Computing), vol. 1042, 2020, pp. 685–696, doi: 10.1007/978-981-32-9949-8_48.

[20] K. He, D. D. Kim, and M. R. Asghar, "Adversarial machine learning for network intrusion detection systems: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 1, pp. 538–566, 1st Quart., 2023, doi: 10.1109/COMST.2022.3233793.

[21] S. M. Kasongo, "A deep learning technique for intrusion detection system using a recurrent neural networks based framework," *Comput. Commun.*, vol. 199, pp. 113–125, Feb. 2023, doi: 10.1016/j.comcom.2022.12.010.

[22] W. Meng, E. W. Tischhauser, Q. Wang, Y. Wang, and J. Han, "When intrusion detection meets blockchain technology: A review," *IEEE Access*, vol. 6, pp. 10179–10188, 2018, doi: 10.1109/ACCESS.2018.2799854.

[23] M. Masdari and H. Khezri, "A survey and taxonomy of the fuzzy signature-based intrusion detection systems," *Appl. Soft Comput.*, vol. 92, Jul. 2020, Art. no. 106301, doi: 10.1016/j.asoc.2020.106301.

[24] Z. Chiba, N. Abghour, K. Moussaid, A. E. Omri, and M. Rida, "A cooperative and hybrid network intrusion detection framework in cloud computing based on snort and optimized back propagation neural network," *Proc. Comput. Sci.*, vol. 83, pp. 1200–1206, 2016, doi: 10.1016/j.procs.2016.04.249.

[25] N. A. Dawit, S. S. Mathew, and K. Hayawi, "Suitability of blockchain for collaborative intrusion detection systems," in *Proc. 12th Annu. Undergraduate Res. Conf. Appl. Comput. (URC)*, Apr. 2020, pp. 1–6, doi: 10.1109/URC49805.2020.9099189.

[26] C. Yenugunti and S. S. Yau, "A blockchain approach to identifying compromised nodes in collaborative intrusion detection systems," in *Proc. IEEE Int. Conf. Dependable, Autonomic Secure Comput., Int. Conf. Pervasive Intell. Comput., Int. Conf. Cloud Big Data Comput., Int. Conf. Cyber Sci. Technol. Congr. (DASC/PiCom/CBDCom/CyberSciTech)*, Aug. 2020, pp. 87–93, doi: 10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00029.

[27] G. Meng, Y. Liu, J. Zhang, A. Pokluda, and R. Boutaba, "Collaborative security," *ACM Comput. Surv.*, vol. 48, no. 1, pp. 1–42, Sep. 2015, doi: 10.1145/2785733.

[28] D. Laufenberg, L. Li, H. Shahriar, and M. Han, "An architecture for blockchain-enabled collaborative signature-based intrusion detection system," in *Proc. 20th Annu. SIG Conf. Inf. Technol. Educ.*, Sep. 2019, p. 169, doi: 10.1145/3349266.3351389.

[29] W. Meng, W. Li, L. T. Yang, and P. Li, "Enhancing challenge-based collaborative intrusion detection networks against insider attacks using blockchain," *Int. J. Inf. Secur.*, vol. 19, no. 3, pp. 279–290, Jun. 2020, doi: 10.1007/s10207-019-00462-x.

[30] H. Benaddi and K. Ibrahimi, "A review: Collaborative intrusion detection for IoT integrating the blockchain technologies," in *Proc. 8th Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, Oct. 2020, pp. 1–6, doi: 10.1109/WINCOM50532.2020.9272464.

[31] E. Androulaki et al., "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, Apr. 2018, pp. 1–15, doi: 10.1145/3190508.3190538.

[32] M. Conti, E. Sandeep Kumar, C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3416–3452, 4th Quart., 2018, doi: 10.1109/COMST.2018.2842460.

[33] A. I. Sanka and R. C. C. Cheung, "A systematic review of blockchain scalability: Issues, solutions, analysis and future research," *J. Netw. Comput. Appl.*, vol. 195, Dec. 2021, Art. no. 103232, doi: 10.1016/j.jnca.2021.103232.

[34] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, "ProvChain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability," in *Proc. 17th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*, May 2017, pp. 468–477, doi: 10.1109/CCGRID.2017.8.

[35] M. Kumar and A. K. Singh, "Distributed intrusion detection system using blockchain and cloud computing infrastructure," in *Proc. 4th Int. Conf. Trends Electron. Informat. (ICOEI)*, Jun. 2020, pp. 248–252.

[36] A. A. Khan, M. M. Khan, K. M. Khan, J. Arshad, and F. Ahmad, "A blockchain-based decentralized machine learning framework for collaborative intrusion detection within UAVs," *Comput. Netw.*, vol. 196, Sep. 2021, Art. no. 108217, doi: 10.1016/j.comnet.2021.108217.

[37] I. Sohn, "Deep belief network based intrusion detection techniques: A survey," *Expert Syst. Appl.*, vol. 167, Apr. 2021, Art. no. 114170, doi: 10.1016/j.eswa.2020.114170.

[38] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116, doi: 10.5220/0006639801080116.

[39] J. Lee, J. Kim, I. Kim, and K. Han, "Cyber threat detection based on artificial neural networks using event profiles," *IEEE Access*, vol. 7, pp. 165607–165626, 2019, doi: 10.1109/ACCESS.2019.2953095.

[40] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems," *Int. J. Eng. Technol.*, vol. 7, no. 3.24, pp. 479–482, Dec. 2018.

[41] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, "Creation of flow-based data sets for intrusion detection," *J. Inf. Warf.*, vol. 16, no. 4, pp. 41–54, 2017.

[42] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, "Flow-based benchmark data sets for intrusion detection," in *Proc. Eur. Conf. Inf. Warf. Secur. (ECCWS)*, 2017, pp. 361–369.

[43] M. Chattopadhyay, R. Sen, and S. Gupta, "A comprehensive review and meta-analysis on applications of machine learning techniques in intrusion detection," *Australas. J. Inf. Syst.*, vol. 22, pp. 1–27, May 2018, doi: 10.3127/ajis.v22i0.1667.

[44] E. K. Reddy, "Neural networks for intrusion detection and its applications," in *Proc. World Congr. Eng.* (Lecture Notes in Engineering and Computer Science), vol. 2, Jul. 2013, pp. 1210–1214.

[45] G. Shen, Q. Tan, H. Zhang, P. Zeng, and J. Xu, "Deep learning with gated recurrent unit networks for financial sequence predictions," *Proc. Comput. Sci.*, vol. 131, pp. 895–903, Jan. 2018, doi: 10.1016/j.procs.2018.04.298.

[46] C. Xu, J. Shen, X. Du, and F. Zhang, "An intrusion detection system using a deep neural network with gated recurrent units," *IEEE Access*, vol. 6, pp. 48697–48707, 2018, doi: 10.1109/ACCESS.2018.2867564.

[47] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, "Light gated recurrent units for speech recognition," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 2, pp. 92–102, Apr. 2018, doi: 10.1109/TETCI.2017.2762739.

[48] T. Lu, Y. Du, L. Ouyang, Q. Chen, and X. Wang, "Android malware detection based on a hybrid deep learning model," *Secur. Commun. Netw.*, vol. 2020, pp. 1–11, Aug. 2020, doi: 10.1155/2020/8863617.

[49] H. M. Lynn, S. B. Pan, and P. Kim, "A deep bidirectional GRU network model for biometric electrocardiogram classification based on recurrent neural networks," *IEEE Access*, vol. 7, pp. 145395–145405, 2019, doi: 10.1109/ACCESS.2019.2939947.

[50] B. Cao, C. Li, Y. Song, Y. Qin, and C. Chen, "Network intrusion detection model based on CNN and GRU," *Appl. Sci.*, vol. 12, no. 9, p. 4184, Apr. 2022, doi: 10.3390/app12094184.

[51] O. Alkadi, N. Moustafa, and B. Turnbull, "A collaborative intrusion detection system using deep blockchain framework for securing cloud networks," in *Proc. SAI Intell. Syst. Conf.*, in Advances in Intelligent Systems and Computing, vol. 1250, 2021, pp. 553–565, doi: 10.1007/978-3-030-55180-3_41.

[52] A. Alshammari and A. Aldribi, "Apply machine learning techniques to detect malicious network traffic in cloud computing," *J. Big Data*, vol. 8, no. 1, pp. 1–24, Dec. 2021, doi: 10.1186/s40537-021-00475-1.

[53] A. Borkar, A. Donode, and A. Kumari, "A survey on intrusion detection system (IDS) and internal intrusion detection and protection system (IIDPS)," in *Proc. Int. Conf. Inventive Comput. Informat. (ICICI)*, Nov. 2017, pp. 949–953, doi: 10.1109/ICICI.2017.8365277.

[54] H. Liu, S. Zhang, P. Zhang, X. Zhou, X. Shao, G. Pu, and Y. Zhang, "Blockchain and federated learning for collaborative intrusion detection in vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 6073–6084, Jun. 2021, doi: 10.1109/TVT.2021.3076780.

[55] O. Alkadi, N. Moustafa, B. Turnbull, and K. R. Choo, "A deep blockchain framework-enabled collaborative intrusion detection for protecting IoT and cloud networks," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9463–9472, Jun. 2021, doi: 10.1109/JIOT.2020.2996590.

[56] M. Keshk, B. Turnbull, N. Moustafa, D. Vatsalan, and K. R. Choo, "A privacy-preserving-framework-based blockchain and deep learning for protecting smart power networks," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5110–5118, Aug. 2020, doi: 10.1109/TII.2019.2957140.

[57] C. Liang, B. Shanmugam, S. Azam, A. Karim, A. Islam, M. Zamani, S. Kavianpour, and N. B. Idris, "Intrusion detection system for the Internet of Things based on blockchain and multi-agent systems," *Electronics*, vol. 9, no. 7, pp. 1–27, Jul. 2020, doi: 10.3390/electronics9071120.

[58] B. Yan and G. Han, "LA-GRU: Building combined intrusion detection model based on imbalanced learning and gated recurrent unit neural network," *Secur. Commun. Netw.*, vol. 2018, pp. 1–13, Aug. 2018, doi: 10.1155/2018/6026878.

EMMANUEL BALDWIN MBAYA received the B.Eng. degree in computer engineering from the University of Maiduguri, in 2014, and the M.Eng. degree in information and communication engineering from Covenant University, in 2022. He is currently a Lecturer with the Department of Computer Science, Federal University, Gashua. He was a Research Assistant with the Covenant Applied Informatics and Communication Africa Centre of Excellence (CApIC-ACE), a World Bank ACE-IMPACT Centre, Covenant University. His research interests include cloud computing, machine (deep) learning, blockchain, and network security.

EMMANUEL ADETIBA (Member, IEEE) received the Ph.D. degree in information and communication engineering from Covenant University, Ota, Nigeria. He was the Director of the Center for Systems and Information Services (aka ICT Center), Covenant University, from 2017 to 2019. He is the incumbent Deputy Director of the Covenant Applied Informatics and Communication Africa Centre of Excellence (CApIC-ACE) and a Co-PI of the FEDGEN Cloud Computing Research Project at the center (World Bank and AFD funded). He is the Founder and a Principal Investigator of the Advanced Signal Processing and Machine Intelligence Research (ASPMIR) Group. He is a Full Professor and the former Head of the Department of Electrical and Information Engineering, Covenant University, from 2021 to 2023. He is also an Honorary Research Associate (HRA) with the Institute of System Sciences and a Visiting Research Associate with the KZN e-Skills Co-Laboratory, Durban University of Technology, Durban, South Africa. He was a recipient of several past and ongoing scholarly grants from reputable bodies, such as World Bank; France Development Agency (AFD); Google; U.S. National Science Foundation; Durban University of Technology, South Africa; Nigeria Communications Commission; Rockefeller Foundation; International Medical Informatics Association (IMIA); and hosts of others. He has authored/coauthored more than 100 scholarly publications in journals and conference proceedings, some of which are indexed in Scopus/ISI/CPCI. His research interests and experiences include machine intelligence, software-defined radio, cognitive radio, biomedical signal processing, and cloud and high performance computing (C&HPC). He is a registered engineer (R.Engr.) with the Council for the Regulation of Engineering in Nigeria (COREN) and a member of the Institute of Information Technology Professional (IITP), South Africa.

JOKE A. BADEJO (Member, IEEE) received the Ph.D. degree in computer engineering from Covenant University, Nigeria, in 2015. She is a Senior Lecturer and a former Coordinator of the Computer Engineering Program, Department of Electrical and Information Engineering, Covenant University. She is also a Faculty Member of the Covenant Applied Informatics and Communication Africa Centre of Excellence (CApIC-ACE), a World Bank ACE-IMPACT Centre, Covenant University. She is a Co-Investigator on a World Bank ACE-IMPACT Project, targeted at developing a federated genomics cloud infrastructure for precision medicine in Africa. She has actively involved with the Covenant University Data Analytics Cluster (CUDAC), which supports data-driven decision-making at the university. With more than a decade of computing research, teaching, and leadership experiences, she has published over 40 papers in reputable journals and conference proceedings. Her broad research experiences and interests include biometrics and biomedical image analysis, machine (deep) learning, large-scale data analytics, software engineering, and cloud computing. She is a registered engineer by the Council for the Regulation of Engineering in Nigeria (COREN). She enjoys being an academic and loves contributing impactful and cost-effective solutions to current societal engineering problems in Africa.

JOHN SIMON WEJIN received the M.Sc. degree in computer engineering from Eastern Mediterranean University, Famagusta, North Cyprus. He is currently pursuing the Ph.D. degree in computer engineering with Covenant University, Ota, Nigeria. He is a Lecturer with the Department of Computer Science, Taraba State University, Jalingo, Nigeria. He is also an Intern with the FEDGEN Cloud Infrastructure Research Project with the Covenant Applied Informatics and Communication Centre of Excellence (CApIC-ACE). He has published eight papers in reputable journals and conferences. His research interests include cloud federation, computer networks, data communication, intelligent systems, artificial intelligence, and data analytics. He is a member of the International Association of Engineers (IAENG).

OLUWADAMILOLA OSHIN (Member, IEEE) received the Ph.D. degree in information and communication engineering (with a focus on nano-electronic biosensing) from Covenant University, Nigeria, in 2020. She is a Lecturer of information and communication engineering with the Department of Electrical and Information Engineering, Covenant University. She is also a Faculty Member of the Covenant Applied Informatics and Communication Africa Centre of Excellence (CApIC-ACE), a World Bank ACE-IMPACT Centre, Covenant University. She has published about 30 papers in reputable journals and conference proceedings. She has broad research experiences and interests including mobile communications, data analytics, and MEMS-based biosensing. She is professionally registered with the Council for the Regulation of Engineering in Nigeria (COREN). She enjoys research and solving health-related issues using engineering and technology.

OLISAEMEKA ISIFE received the degree in computer engineering from Federal University Oye Ekiti, in 2018, and the master's degree in computer engineering from Covenant University, in 2023. His major field of study is computer engineering. Previously, he has published papers in various areas, such as embedded systems application and off-grid power solutions utilizing solar energy. His current research is primarily focused on cybersecurity, software development, and deep learning.

SURENDRA COLIN THAKUR is currently an Associate Professor with the Department of Information Technology, Durban University of Technology (DUT), Durban, South Africa. He is also the Director of the NEMISA KZN e-Skills Co-Laboratory, DUT, which is tasked with e-skills in general, and particularly on the e-enablement of government services for effective service delivery, with a focus on e-democracy and e-participation. He has also conceptualized and introduced InvoTech, an innovation incubator with DUT, where one of his patents is being registered. He is an international e-voting expert, an emerging expert in social media, a national authority on big data, and the Director of the KZN Digital Co-Laboratory, DUT. His research interests include data science, e-voting and public participation, and social media. He has served on many executive forums over the years, such as executive committee (exco), senate, and faculty board. He was also the National Treasurer of the Computer Society of South Africa (CSSA) for a year and the Chair of the KZN Computer Society of South Africa for three years and the Vice-Chairperson for two years.

SIBUSISO MOYO received the Ph.D. degree in mathematics with a focus on symmetries of differential equations and their application from the University of Natal (currently University of KwaZulu-Natal), Durban, and the master's degree in tertiary education management from the University of Melbourne. She is currently the Deputy Vice-Chancellor of research, innovation and postgraduate studies with Stellenbosch University, South Africa. She has published widely in the *Mathematical Sciences*. Her Google profile can be found here: https://scholar.google.com/citations?user=s04NCusAAAAJ&hl=en.

EZEKIEL F. ADEBIYI is a H3Africa, DFG (aka German Science Foundation) Projects Principal Investigator and the Head of the Covenant University Bioinformatics Research (CUBRe) Group. He was the Centre Leader for the Covenant Applied Informatics and Communication African Centre of Excellence (CApIC-ACE), a new World Bank funded ACE Impact Project, Covenant University.

• • •