Research Article

Wisam Elmasry*, Akhan Akbulut, and Abdul Halim Zaim

A Design Of An Integrated Cloud-based Intrusion Detection System With Third Party Cloud Service

https://doi.org/10.1515/comp-2020-0214
Received Jul 13, 2020; accepted Nov 23, 2020

Abstract: Although cloud computing is considered the most widespread technology nowadays, it still suffers from many challenges, especially related to its security. Due to the open and distributed nature of the cloud environment, this makes the cloud itself vulnerable to various attacks. In this paper, the design of a novel integrated Cloud-based Intrusion Detection System (CIDS) is proposed to immunise the cloud against any possible attacks. The proposed CIDS consists of five main modules to do the following actions: monitoring the network, capturing the traffic flows, extracting features, analyzing the flows, detecting intrusions, taking a reaction, and logging all activities. Furthermore an enhanced bagging ensemble system of three deep learning models is utilized to predict intrusions effectively. Moreover, a third-party Cloud-based Intrusion Detection System Service (CIDSS) is also exploited to control the proposed CIDS and provide the reporting service. Finally, it has been shown that the proposed approach overcomes all problems associated with attacks on the cloud raised in the literature.

Keywords: Cloud Security, Deep Learning, Intrusion Detection, Particle Swarm Optimization, Ensemble System

1 Introduction

In recent years, cloud computing is deemed to be the most emerging Internet-based technology in the Information Technology (IT) world. It first started when Google introduced the concept of cloud computing in 2006. Thereafter, it grew rapidly until it became the preferred choice of every individual and organization involved in the IT industry due to the revolution that it made in IT world with its new computing and communication paradigms. For instance, it offers dynamically scalable and virtualized resources which can be quickly provisioned and released with worry-free management effort from the users. Further, these shared resources such as storage, information, platforms, applications, and servers are provided to the cloud users as services on demand. Therefore, the users only pay for resources that they use. Accordingly, this pay-per-use manner makes it possible for many business organizations to avoid capital expenditure for traditional IT which they might not able to afford [44].

Basically, cloud computing consists of three abstract layers, namely, system layer, platform layer and application layer. Whereas, the first two layers related to the Virtual Machines (VM) and operating system, the last one includes applications provided by the cloud such as web-based applications. Moreover, the services are provided to the users in three different models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). IaaS targets the administrators for full controlling and maintaining of the VMs and network. Meanwhile, PaaS model targets the developers to deploy user-created applications in the cloud, SaaS enables users to execute provider's applications. The cloud architecture comprises of two sequential components. The first is known as front-end which is the part seen by the user, *e.g.* the user's network or computer, and the application that used to access the cloud via a user-interface. The second component is the back-end which is the cloud itself and it consists of various servers [46].

Along with the benefits and popularity of cloud computing, it suffers from serious obstacles which act as barriers of its success. One of which is the cloud security that is considered as a major hurdle of cloud adoption by most of organizations. This due to the nature of the cloud environment which is open and fully-distributed, hence making it more prone to security threats and vulnerabilities. Thus, this encourages the intruders to launch potential attacks against the cloud or against devices inside the cloud. An-

***Corresponding Author: Wisam Elmasry:** Department of Computer Engineering, Istanbul Commerce University, 34840, Istanbul, Turkey; Email: wisam.elmasry@istanbulticaret.edu.tr
Akhan Akbulut: Department of Computer Engineering, Istanbul Kultur University, 34158, Istanbul, Turkey;
Email: a.akbulut@iku.edu.tr
Abdul Halim Zaim: Department of Computer Engineering, Istanbul Commerce University, 34840, Istanbul, Turkey;
Email: azaim@ticaret.edu.tr

other vulnerability of cloud security is that the users access their data on remote servers in the cloud with no responsibilities of data storing and maintaining which is done solely by the Cloud Service Provider (CSP). This relinquishing of the control over data and applications poses critical concerns of data integrity, confidentiality and availability [17].

Cloud security is an active area of research and many solutions are proposed and evolved. The traditional cloud architecture opens doors for the risk of a cloud intrusion. An intrusion is defined as an attempt to compromise Confidentiality, Integrity and Availability (CIA) of a computer, network or cloud. This can be handled by executing one or more of intrusive attacks. There are many of attacks that can take place in cloud computing, *e.g.*, Distributed Denial of Service (DDoS), flooding, and port scanning. One of the accurate mechanisms to prevent such attacks is using a cloud-based intrusion detection system [31].

CIDSs vary depending on their type and the detection method that they use. There are three famous types of CIDS, namely, Host-based Intrusion Detection System (HIDS), Network-based Intrusion Detection System (NIDS), and Distributed Intrusion Detection System (DIDS). HIDS runs on a specific host machine to monitor and analyze collected data to detect intrusive events. NIDS identifies intrusions on key network points by monitoring and capturing all traffic on the entire network. DIDS uses both HIDS and NIDS. Regarding detection methods, in the open literature, there are three well-known techniques, namely, signature-based detection, anomaly-based detection, and hybrid detection. Signature-based detection or also so-called misuse detection identifies intrusions by matching the collected data with a database of patterns of renowned attacks or predefined set of rules. The main drawback of the latter technique is that it can only detect known attacks. On the other hand, anomaly-based detection detects intrusions by comparing the collected data against an established baseline. If the current activity deviates from the normal case, then it raises an alarm of a possible malicious attack. The main advantage of anomaly-based detection is that it can detect either known or unknown attacks. Hybrid detection combines signature-based and anomaly-based together in one process. It was reported that CIDS which use hybrid detection achieve better results than in other techniques [51].

In the literature, there are many CIDSs that have been proposed to detect intrusion in cloud computing. Despite of this, most of existing CIDSs have some limitations such as the ability to adapt with a high rate access in the network of the cloud, vulnerability of detecting new and distributed attacks, and lack of control from users [26]. Therefore, there is a need to develop a reliable CIDS taking into account all these issues. The aims and contributions of this research are four-fold:

- In this paper, we extend our former work [19–22] in masquerade detection and network intrusion detection to study the concerns and challenges in cloud intrusion detection area.
- An integrated cloud-based intrusion detection system is proposed using an ensemble system and meta-heuristics.The proposed CIDS can monitor the entire network of the cloud, detect possible intrusions, and log all events.
- The proposed CIDS is superior to the existing CIDSs in the literature in terms of performance, security, robustness, consistency and stability.
- To provide transparency, a third-party cloud service is introduced at the top of the cloud.

The rest of this paper is organized as follows. A summary of literature review is presented in Section 2. In Section 3, a set of essential concepts, processes and algorithms are introduced. Section 4 then explores the usage of ensemble system and which models are used. Meanwhile Section 5 presents the design of the proposed CIDS, Section 6 describes the functionality of the proposed CIDSS. Finally, Section 7 generalises the analysis and advantages of the proposed CIDS and CIDSS. Conclusions are drawn in Section 8.

2 Related Work

There are many of previous studies on the field of cloud intrusion detection. However, in this study, we focus only on the design of CIDSs and services rather than the detection method used. To start with, Yee *et al.* designed a CIDS to protect web services against certain attacks [68]. The proposed web service cannot be controlled by the cloud users. A model for new generation of CIDS is introduced by Bosin *et al.* [8]. They proposed a web service to enable users to access intrusion detection services whenever needed. Moreover, their web service does not allow the cloud users to control the entire CIDS.

Vieira *et al.* proposed a CIDS at cloud middleware layer for capturing data packets and then analyzing them using a hybrid CIDS [64]. They simulated their proposed CIDS and found its performance acceptable for real-time cloud environment. Despite that, they did not include a reporting mechanism to the cloud users within their CIDS. In the study of Roschke *et al.* [55], they introduced a central CIDS management that cooperates with a web service based on the VM-based CIDS proposed by Laureano *et al.* [37]. Their framework used many sensors for capturing data from different cloud layers. Then the audit data is stored in a database and analyzed by the analysis component. If the central CIDS finds an attack, it will notify the user and give him a full control to monitor and administer the CIDS. A distributed system for intrusion detection in clouds using mobile agents is proposed by Dastjerdi *et al.* [15]. It was located at each VM to detect known and unknown attacks in real-time. In spite of that, their system has a weakness point on the number of VMs to be visited.

Lo *et al.* proposed a cooperative Intrusion Detection System (IDS) framework for cloud computing networks [39]. Their distributed framework comprised of two separated components which are a web service and an IDS. The web service is designed to immunise the cloud from DDoS attacks. On the other hand, the IDS is implemented according to the studies of [52, 59]. Mazzariello *et al.* emplaced a NIDS on virtual switch of the physical machine that hosting user virtual machines [42]. Their NIDS proved its efficiency in terms of load sharing of large volumes of data. An IDS in VMs for securing cloud from DDoS attacks is introduced in the study of [6]. It was a NIDS that can detect only known DDoS attacks efficiently.

Patel *et al.* proposed an autonomic agent-based self-management cloud Intrusion Detection and Prevention System (IDPS) to detect most types of attacks in real-time [50]. Although they did not give any details about the implementation of their work. A distributed CIDS model is introduced by Gul *et al.* in which it can handle a large amount of traffic flow by using multithreading [26]. Further, they implemented a third-party CIDSS that is responsible for monitoring and sending information to the cloud user as well as an expert advice to the CSP. Lee *et al.* designed a multi-level HIDS and log management at each guest operating system in cloud computing [38]. Their hybrid HIDS can detect attacks at fast rate but consumes more resources for high level users as a limitation.

Furthermore, one of the significant researches on the field is the work of Kholidy *et al.* who developed a framework for a CIDS. Their framework is scalable and elastic with no central coordination which can detect threats in a hybrid manner [32]. If an attack is detected, then the system alerts the CSP by a report. Alsafi *et al.* introduced an effective and efficient IDPS which combines both IDS and Intrusion Prevention System (IPS) in one mechanism [2]. The proposed IDPS is a hybrid detection system that can detect a various number of attacks and stop them. A scalable IDS is developed based on cloud computing by Zarrabi *et al.* [69]. It benefits from cloud computing features and it also overcomes the cyber attacks. In the study of Shaikh *et al.* [56], a trust-based framework was proposed for achieving security within cloud environments. The proposed framework depends on calculating a trust value which can be utilized as a security strength evaluator for both CSP and users. Shelke *et al.* introduced a multi-threaded distributed CIDS [58]. It uses a hybrid detection method to detect attacks within cloud and sends reports to the users by a third party cloud service.

Kholidy *et al.* proposed a hierarchical, autonomous, and forecasting-based CIDS that can monitor, analyze the system events, and evaluate the risk level [33]. It is a NIDS that depends on a forecasting engine that runs the Holt-Winters algorithm. Another work by Kholidy *et al.* presented three attack prediction models for their proposed Autonomic Cloud Intrusion Detection Framework (ACIDF) [34]. They examined three models, namely, Finite Context Prediction Model (FCPM) that uses a Variable Order Markov Model (VMM) with a Probabilistic Suffix Tree (PST), Finite State Hidden Markov Prediction Model (FSHMPM), and Holt-Winters Prediction Model (HWPM). They reported that the examined models scored good results when validated on a DARPA dataset [45]. A detection method for illegal access for cloud environment is proposed by Alguliev *et al.* [1]. The proposed anomaly-based detection method detected any abnormal behavior in the cloud by using the cosine similarity method and applying the collaborative filtering method. In the study of Wang *et al.* [67], they designed a Cloud Service Trust Evaluation Model (CSTEM). The proposed model based on combining weights and gray correlation analysis.

Finally, it can be observed from the above reviewed literature that some studies concentrated on solving the problem of handling a large flow of cloud data by proposing a multithreaded CIDS. Whilst others tried to design a third party cloud service to deal with cloud users when an attack occurred. The rest introduced hybrid detection IDS to immunise the cloud against known and unknown attacks. Therefore, the design of an integrated CIDS which is multithreaded, hybrid and dynamic with a third party user-friendly cloud service is still a big challenge.

3 Materials and methods

This section introduces all preliminary terms and processes that are vital to our design of CIDS.

3.1 Feature extraction

Machine learning-based security approaches are likely vulnerable to poisoned datasets which can be caused by a legitimate defender's misclassification or attackers aiming to evade detection by contaminating the training data set. There also exist obvious gaps between the lab environment and the real world. Giving importance to the former fact which necessitates the need for an auditing system as a part of any cloud-based IDS. Regardless of the type of detection method, the auditing system captures data flow passing through the entire infrastructure of the cloud. Then, the collected flow should be analyzed by carrying out an important process called feature extraction. Afterwards, these extracted features are useful for training or building prediction models.

Basically, the term "feature" stands for a particular aspect of information in the record. Thus, feature extraction is a process where a set of descriptive statistics of the traffic flow is extracted for each packet separately. After that, these extracted features are saved together into one record where each field of that record represents one of the extracted features. Continuing in this manner, a set of records of different data packets are obtained and stored in a database or data warehouse for training or testing purposes. Determining which features to be extracted from the collected raw data is a big dilemma. Because it is critical to extract only features that are able to characterize successfully the activity of the origin data packets. This will help prediction models to determine whether the packet has a malicious activity or not, and then a decision is taken to decide which traffic may pass and which traffic is blocked accordingly. In this subsection, we specify a set of features to be extracted by the auditing system of our proposed CIDS. These features are well-known and are reported in some of previous studies in the literature.

Lashkari *et al.* [36] presented a new software known as *CICFlowMeter* which is publicly available on the Canadian Institute for Cybersecurity website [12]. By using *CICFlowMeter*, they generated *Tor* flows and proposed a set of time-related features to identify and characterize *Tor* traffic. In addition to that, they have proved that using time-related features only they can identify and characterize *Tor* traffic to some extent.

After that, Sharafaldin *et al.* [57] have extended the former study and published a new reliable IDS dataset called Canadian Institute for Cybersecurity Intrusion Detection Systems 2017 (CICIDS2017) [9]. CICIDS2017 is completely labeled dataset and has 80 network traffic features which are extracted from *pcap* raw data using the CICFlowMeter. Further, the features are extracted and calculated for both benign and intrusive flows which are based on various applications and network protocols. In order to cover a diverse set of common attack scenarios, they executed 20 different attack types which can be categorized into 7 major attack families or categories, namely, Brute Force, Heartbleed, Botnet, Denial of Service (DoS), DDoS, Web attack, and Infiltration. Moreover, they tested the full extracted features using a RandomForestRegressor to select a short feature subset which can be the best detection features for all seven attack categories. Hence, they reported that a subset of 23 features is the optimal feature set for all attack types [57]. The list of the optimal features and the corresponding attack categories and protocols is shown in the first row of Table 1. The Forward term means a traffic flow from the source to the destination. Whereas, the Backward term indicates a traffic flow in the inverse direction, that is, from the destination to the source. Finally, the term Flow means a bi-directional flow in either direction.

In a similar manner, the Coburg Intrusion Detection Data Set (CIDDS) is a labeled flow-based dataset for the evaluation of anomaly-based IDS, and it is also publicly available on the Internet [13]. Indeed, CIDDS has two versions, namely, CIDDS-001 [53] and CIDDS-002 [54]. The main difference between them is that CIDDS-001 has various attack types with a total of 92 attack types whilst CIDDS-002 is a port scan attack only dataset with a total of 43 port scan attacks. Furthermore, they collected unidirectional *NetFlow* audit data and analyzed them to extract 10 different features as well as an additional attribute for the appropriate label. The attack types in CIDDS-001 are also grouped into four major attack categories such as DoS, Brute Force, Port Scan, and Ping Scan. Thus, we included nine features of CIDDS to our auditing system, where the remaining feature is duplicated. The selected features of CIDDS are presented in the second row of Table 1.

Moreover, NSL-KDD is a well-known IDS dataset and widely used in intrusion detection field [49, 61]. NSL-KDD is an improved version that resolved all limitations of earliest IDS datasets: DARPA [43, 45] and KDD CUP 99 [60, 62]. They captured *tcpdump* raw data and then processed it to extract 41 different features. In addition to that, NSL-KDD has 38 attack types which are combined into four major attack categories, namely, DoS, Probing (Probe), Remote to Local (R2L), and User to Root (U2R). There are many previous studies that performed a feature selection process to NSL-KDD to select feature subsets that suited with each attack category [3, 5, 47, 48, 66]. We concluded all these feature subsets in the third row of Table 1 after removing duplicated attributes.

In this study, we selected all the forty features in Table 1 to be calculated and extracted from traffic flows through our auditing system by using the same procedures that were used in the previous studies discussed above. We believe that the selected set of features are able to represent the different activities of the users as well as cover wide range of common attack families. Finally, Gharib *et al.* [25] proposed an evaluation framework for IDS datasets that reflects the characteristics of a valid dataset, and consists of eleven characteristics, namely, Complete Network Configuration, Complete Traffic, Labeled Dataset, Complete Interaction, Complete Capture, Available Protocols, Attack Diversity, Anonymity, Heterogeneity, Feature Set, and Metadata. It has been shown that the selected features in Table 1 are

Table 1: List of extracted features and the corresponding attack families and protocols.

Features	Attack Categories	Protocols
Backward Packet Length Minimum	Brute Force	HTTP
Subflow Forward Bytes	Heartbleed	HTTPS
Total Length Forward Packets	Botnet	FTP
Forward Packet Length Mean	DoS	SSH
Backward Packet Length Standard deviation	DDoS	Email protocols
Flow Inter Arrival Time Minimum	Web Attack	
Forward Inter Arrival Time Minimum	Infiltration	
Flow Inter Arrival Time Mean		
Flow Duration		
Flow Inter Arrival Time Standard deviation		
Active Minimum		
Active Mean		
Backward Inter Arrival Time Mean		
Initial Window Forward Bytes		
Acknowledge Flag Count		
Forward Push Flags		
Synchronization Flag Count		
Forward Packets/second		
Initial Window Backward Bytes		
Backward Packets/second		
Push Flag Count		
Average Packet Size		
Forward Inter Arrival Time Mean		
Source IP Address	DoS	TCP
Source Port	Brute Force	UDP
Destination IP Address	Port Scan	ICMP
Destination Port	Ping Scan	
Transport Protocol		
Start Time Flow First Seen		
Number of Submitted Bytes		
Number of Transmitted Packets		
TCP Flags		
Network Service	DoS	SMTP
Number of Received Bytes	Probe	HTTP
Source Connections Count	R2L	FTP
Synchronization Flag Error Rate	U2R	Telnet
Different Services Rate		ICMP
Destination Connections Count		SNMP
Reject Flag Error Rate		
Rate of connection to different destinations		

totally complied with aforementioned evaluation framework [25].

3.2 Data preprocessing

Most of the machine learning models can only work with numerical values for training and testing. Therefore, it is necessary to convert all non-numerical values to numerical values by performing data numericalization. We prefer to use the method of data numericalization such that for each nominal feature, its values are ordered alphabetically. After that, the ordered nominal values are converted to numerical values by assigning specific values to each variable ranged in [1, length of the list].

Then, the data normalization process has taken place when all numeric features in the dataset (including the transformed nominal features) are mapped into [0,1] linearly by using the following Min-Max transformation formula [65].

$$x_i = \frac{x_i - Min}{Max - Min}, \tag{1}$$

where x_i is the numeric feature value of the i^{th} sample, Min and Max are the minimum and maximum values of every numeric feature, respectively.

3.3 Feature and Hyperparameter selection

Recently, Elmasry *et al.* [21] proposed a novel double Particle Swarm Optimization (PSO) algorithm for automatic feature and hyperparameter selection. It is a top-down hierarchical metaheuristic which simply consists of two sequential levels. In the upper level, it receives the given dataset with full feature set (D) and runs the Fitness Proportionate Selection Binary Particle Swarm Optimization and Entropy (FPSBPSO-E) algorithm [21] to select the optimal feature subset. FPSBPSO-E is a modified version of the single-objective filter-based feature selection method that was introduced by Cervante *et al.* [10]. The major difference between FPSBPSO-E and the method in the study of [10] is applying the Fitness Proportionate Selection Binary Particle Swarm Optimization (FPSBPSO) algorithm [70] in the core of optimization process rather than using the traditional Binary PSO. FPSBPSO-E determines the relevance and redundancy of the selected feature subset by measuring the entropy of each group of features. Mathematically, the fitness function can be computed using the following formula [10].

$$Fitness_2 = \alpha_2 \times D_2 - (1 - \alpha_2) \times R_2 \tag{2}$$

where

$$D_2 = \sum_{c \in C} IG(c|X),$$

$$R_2 = \frac{1}{|S|} \sum_{x \in X} IG(x|\{X/x\}),$$

where X is the set of the selected features and C is the set of class labels. D_2 indicates the relevance between the selected features and the class labels by calculating the information gain (entropy) in the class labels given information of the selected features. R_2 evaluates the redundancy contained in the selected feature subset by measuring the joint entropy of all the selected features. $Fitness_2$ is a maximization fitness function which minimizes the redundancy (R_2) and simultaneously maximizes the relevance (D_2). In addition to that, α_2 is a weight parameter which is a constant value ranged in [0,1]. Finally, FPSBPSO-E outputs the optimal feature subset and reduces the original dataset to the reduced dataset D^* with only the selected features.

After that, in the lower level, the double PSO-based algorithm receives a copy of the reduced dataset D^* and the type of the deep learning model M as inputs and then executes a continuous PSO-based algorithm [19] for hyperparameter selection. The PSO-based algorithm maximizes the accuracy of model M over the given training set of the reduced dataset D^* and outputs the optimal hyperparameters vector H^* of the model M. Finally, the double PSO-based algorithm outputs both D^* and H^* and terminates. Figure 1 shows the diagram of the double PSO-based algorithm.

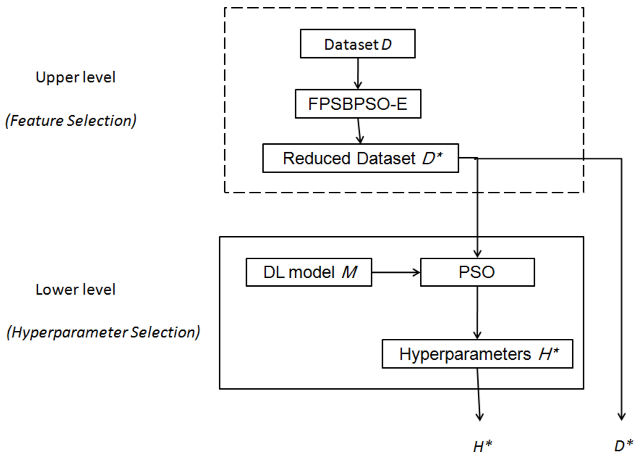

Figure 1: The diagram of the double PSO-based algorithm.

4 Ensemble system

Ensemble system or also known as ensemble learning is a technique where a combination of several classifiers is used with the aim of creating an improved composite model [14]. The structure of an ensemble system can be based either on classifiers of different types (heterogeneous ensemble) or of the same type (homogenous ensemble) [24]. The main merits of using such ensemble system are to increase accuracy and robustness, better overall generalization, and ability to handle large data. The main idea behind the ensemble system is to train all combined classifiers simultaneously on a training set. Then, testing every unseen data against a series of the learned classifiers in such a way that each classifier produces its decision. Later, their decisions are fused together in order to obtain the final decision. In the literature, there are two methods to combine all decisions together, namely, bagging method and boosting method. The bagging method combine all decisions by averaging the prediction over a collection of classifiers. Whereas, the boosting method weighted vote with a collection of classifiers.

Recently, exploiting ensemble system in intrusion detection area has witnessed a surge of research efforts [7, 11, 18, 30, 40, 41]. The experimental results of the previous researches indicated that the ensemble-based approach reaches more reliable results when compared with the single classifier. In this study, a bagging ensemble system is utilized to detect intrusions in the proposed CIDS. The proposed bagging ensemble system consists of two adjacent parts as shown in Figure 2.

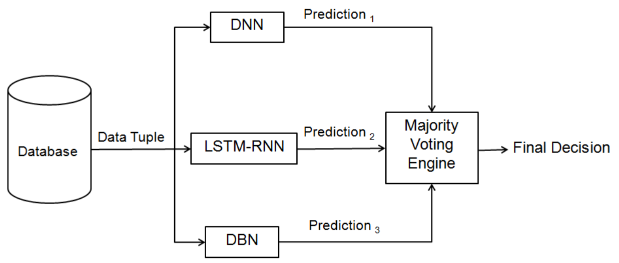

Figure 2: The basic structure of the proposed Bagging ensemble system.

The first part comprises of a series of three deep learning models, Namely, Deep Neural Networks (DNN) [27], Long Short-Term Memory Recurrent Neural Networks (LSTM-RNN) [29, 35], and Deep Belief Networks (DBN) [28]. The reasons for choosing these models rather than other deep learning models are: i) The experimental findings in our previous studies [19–21] confirmed their effectiveness in both masquerade detection and network intrusion detection; ii) many review articles pointed out their success in solving intrusion detection problem [4, 63]; iii) they are common in the static classification tasks such like intrusion detection. On the other hand, the second part is merely a majority voting engine to fuse all votes of the individual models to the final decision depending on majority (the most votes). Table 2 presents the truth table of the majority voting engine. Notably, the majority is guaranteed when taking the final decision because we have an odd number of models (three). Finally, we believe that building such a bagging ensemble system is worthwhile in order to compensate for the mistakes of a single deep learning model by the other deep learning models.

Table 2: The truth table of the majority voting engine.

Predictions			Final decision
DNN	**LSTM-RNN**	**DBN**	
Normal	Normal	Normal	Normal
Normal	Normal	Attack	Normal
Normal	Attack	Normal	Normal
Normal	Attack	Attack	Attack
Attack	Normal	Normal	Normal
Attack	Normal	Attack	Attack
Attack	Attack	Normal	Attack
Attack	Attack	Attack	Attack

5 Proposed CIDS

The open and distributed nature of cloud computing as well as its service oriented paradigm, make cloud infrastructure highly vulnerable to various potential attacks. The traditional IDSs are insufficient for such environment and threats. Therefore, there is a real need for developing an integrated cloud-based IDS which takes into account all the emerging problems and challenges during its design. An integrated CIDS means that incorporating renowned solutions to communicate over a single platform. In this section, we propose the design of our integrated cloud-based IDS. Figure 3 depicts the flowchart of the proposed cloud-based IDS. It consists of five main modules, namely, monitoring, processing, analysis, prediction, and response. The details of each module are discussed in the upcoming subsections.

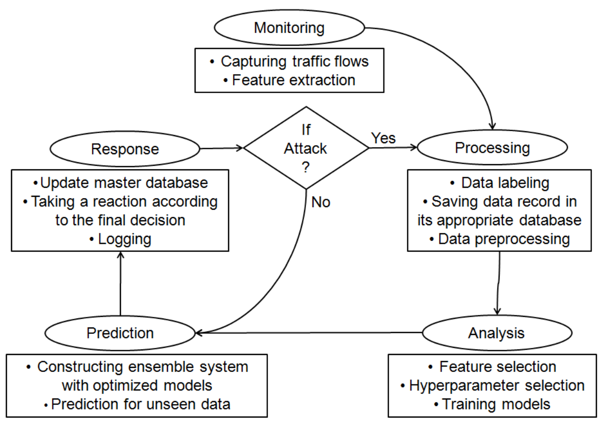
Monitoring
• Capturing traffic flows
• Feature extraction
If Attack ?
Yes
No
Response
• Update master database
• Taking a reaction according to the final decision
• Logging
Processing
• Data labeling
• Saving data record in its appropriate database
• Data preprocessing
Prediction
• Constructing ensemble system with optimized models
• Prediction for unseen data
Analysis
• Feature selection
• Hyperparameter selection
• Training models

Figure 3: The flowchart of the proposed CIDS.

5.1 Monitoring module

Primarily, the cloud users access their data and applications on remote servers at the site of CSP over the cloud network. Thus, user actions and requests must be monitored and logged through an auditing system. The monitoring module is deemed to be the auditing system of the proposed CIDS. It performs two separated functions, namely, capturing traffic flows and feature extraction.

The first function of the monitoring module is capturing all the in-bound and out-bound data packets that are traversing within the cloud network. In order to facilitate this task, the monitoring module uses sensors to sensitize network traffic. Further, the monitoring module has the flexibility to capture data packets of different application, transport and network protocols such like TCP, UDP, ICMP, IP, HTTP, SMTP, etc. The collected traffic flows is stored immediately in the shared queue. The aim of using the shared queue is to be an intermediate station between packets capturing and feature extraction, that is, to store collected data packets until the feature extractor processes them sequentially. The shared queue has enough space in term of storage capacity to store the collected data packets without overflow.

Afterwards, the feature extraction process takes place through a feature extractor. Our feature extractor handles every data packets in the shared queue in such a way it extracts 40 features from the data packet as mentioned in subsection 3.1. Then, it put all the extracted features of the data packet into a data record or tuple and sends it to the next module, *i.e.*, the processing module. These data records are the essence of the scene and the raw material that we need for training and prediction. Figure 4 shows the process of the monitoring module. It is worthy to say

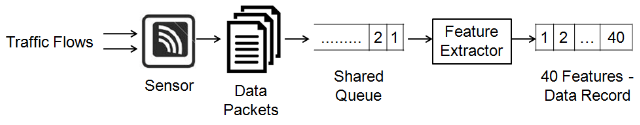

Figure 4: The process of the monitoring module.

that the operation of the monitoring module is lasting, continuous and independent from other modules.

5.2 Processing module

The processing module performs all required missions prior to the prediction process. It firstly receives data records from the feature extractor and tries to assign them to the proper class label (either “Normal” or “Attack”). This can be accomplished by leveraging a signature-based detection procedure which employs a set of pre-defined rules (signatures) to match the inspected data record against known attack patterns. If there is a match, the data record will be labeled as an “Attack”. Otherwise, it will be labeled as “Normal”. These rules are defined and set by the cloud service provider. After that, processing module stores the labeled data record in a database we named it as “master database”.

The former process will take place only in the first run of the system. In other words, in the first run, the processing module will create the master database from scratch, receive data records, label data records using the above mentioned method, and save the labeled data records in the master database sequentially. Once the master database becomes rich enough in diversity and quantity, the processing module will stop the former process and creates another database from scratch. We named it as “unseen database”. Then, all new coming data records will be saved in the unseen database without labeling. Accordingly, the master database which contains labeled data records will be used as the training set of our cloud-based IDS. On the other hand, the unseen database which contains unlabeled data records will be the test set.

The last function of the processing module is data preprocessing. It occurs just when processing module finished filling in the master database. The data preprocessing will be handled on the master database using the same procedure which is explained in subsection 3.2.

5.3 Analysis module

The analysis module performs pre-training and training phases of the deep learning models. In the pre-training phase, the analysis module executes the double PSO-based algorithm which is explained in subsection 3.3 for both feature and hyperparameter selection. Firstly, the analysis module receives a copy of the master database after data preprocessing. Next, it executes the upper level of the double PSO-based algorithm on the master database to find out the optimal feature subset. After that, it reduces the master database using only the optimal features. Then, it executes the lower level of the double PSO-based algorithm by using the reduced master database to obtain the optimal hyperparameter vector of the desired deep learning model. Because we have three deep learning models, the latter step will be repeated three times one for DNN, LSTM-RNN and DBN models, respectively. Afterwards, the training phase takes place for each deep learning model separately on the reduced master database.

5.4 Prediction module

The prediction module is the core of the proposed CIDS that performs two sequenced functions. The first is construction of the bagging ensemble system which was proposed in Section 4 using the learned DNN, LSTM-RNN and DBN models that previously obtained from the analysis module. Then, processing module gets a copy of the unseen database and performs a data preprocessing on it as well. The second function of the prediction module is to pick up a data record from the preprocessed unseen database sequentially. After that, the picked data record is tested using the bagging ensemble system, and the final decision obtained from the majority voting engine is delivered to the response module.

5.5 Response module

The response module is in charge of controlling the whole operation in the proposed CIDS by carrying out three major functions. First, it receives the final decision along with the tested data record, and then it labels the tested data record with the final decision. Thereafter, it updates the master database by saving the new labeled data record at the end of the master database.

The second function is responsible for taking a reaction upon the final decision whether it is “Normal” or “Attack”. In the case of “Normal”, the response module sends a command to the prediction module to continue with the next data record in the unseen database to predict its label. On the other hand, if the received final decision is “Attack”, then the response module raises an alarm that the cloud network is undergoing a potential malicious activity. Afterwards, the response module terminates the session of the attacker as well as blocks all traffic flows belonging to that malicious activity. In addition to that, the response module enforces the processing module to perform a data preprocessing on a new copy of the updated master database, and sends the preprocessed master database to the analysis module. Then, the analysis module runs again to produce new optimized deep learning models, and sends them to the prediction module which in turn constructs the bagging ensemble system again and predicts a new data record from the unseen database.

The final function of the response module is logging all events in a private log file for further processing later. Figure 5 depicts the framework of the proposed cloud-based IDS.

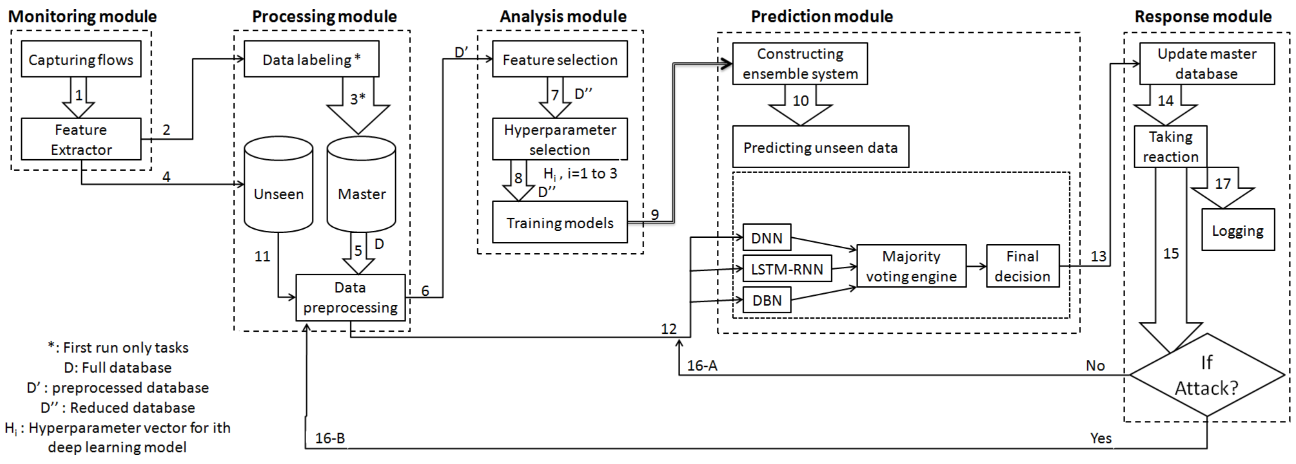

Figure 5: The framework of the proposed CIDS.

6 Proposed CIDSS

Another shortcoming of the traditional IDSs is the lack of control of the user because usually the traditional IDSs are monitored and administrated by the cloud service provider. Thus, if an intruder succeeds in penetrating and stealing or damaging the data of a user, the user will not be notified directly. Instead of that, the IDS will send all information about intrusion to the CSP, and the cloud user has to rely on that. For the sake of image and reputing, the CSP might not inform the cloud user about the loss and conceals the information. In such a scenario, a neutral cloud service can ensure adequate alerting and monitoring for cloud users.

Giving importance to the former case, we propose a third party cloud service that can monitor the proposed CIDS as well as provide alert reports to both the cloud users and CSP. Indeed, it is a web-based cloud service that has two main functions, namely, administration and reporting. The administration function means that the proposed cloud service can deal with both the user and CSP to allow them to configure the proposed CIDS with certain privileges. On the other hand, in the case an attacker manages to perform a malicious activity on the user's data or applications, the proposed cloud service receives an alert from the response module with recent information in the log file. Hence, the proposed cloud service in turn prepares a detailed report and sends it directly to the cloud user and CSP as well. Figure 6 depicts the main functions of the third party cloud service. We believe that by employing the proposed CIDSS, both the transparency of information and security of data can be achieved.

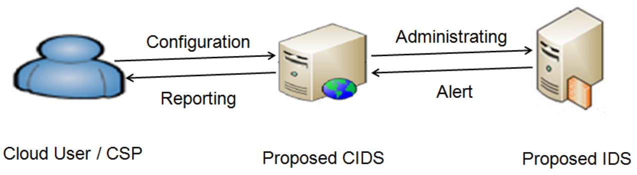

Figure 6: The main functions of the proposed CIDSS.

The type and location of IDS within the cloud infrastructure are very critical issues that arose when designing a cloud-based IDS. Basically, the cloud computing uses the concept of "virtualization" of resources, where many cloud users are hosted on one physical machine "hypervisor server" in a cloud data center. Deploying the CIDS as a HIDS in hypervisor will allow the CSP to monitor the VMs on that hypervisor. With high volumes of traffic and data flow, there would be data packets dropping and overloading of the VM that is hosting CIDS. Moreover, if an offending intruder compromised the host machine, the HIDS that is hosted on that host machine will be neutralized. Therefore, deploying the CIDS as a NIDS will be the best choice in such a cloud environment [26].

Accordingly, we designed the proposed CIDS to be a NIDS and placed it at the back-end of the cloud infrastructure for internal and external intrusion detection. More specifically, the proposed CIDS will be placed outside the VM servers on a key network choke point such as switch at the site of the cloud servers. This global view of the system will enable the proposed CIDS to monitor network traffic efficiently. In addition to that, we placed the proposed CIDSS at the top of the cloud infrastructure (directly after the front-end of the cloud) on a bottle neck of network points such as the router or gateway. Figure 7 shows the architecture of the proposed CIDS and CIDSS.

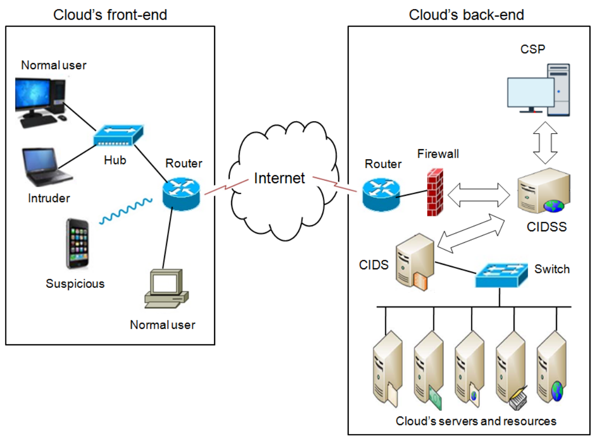

Figure 7: The architecture of the proposed CIDS and CIDSS.

As we can seen from Figure 7, placing the proposed CIDSS as a SaaS at the top of the cloud as well as placing the proposed CIDS as a NIDS at central point in the back-end of the cloud makes our design as a distributed approach across all regions of the cloud.

7 Analysis and Discussion

In this section, we present the advantages of the proposed cloud-based IDS, and how our CIDS resolves the limitations of the existing cloud-based IDSs that were discussed in Section 2.

7.1 Hybrid CIDS

It is reported that the traditional IDSs are not suitable for cloud environment, because the signature-based detection IDSs cannot detect new or unknown attacks, meanwhile, the anomaly-based detection IDSs are computationally expensive and need a large amount of pure normal data [46]. As mentioned in Section 5, the proposed CIDS combines the two detection techniques together, since it performs knowledge-based detection (known trails of previous attacks) in the processing module as well as performs behavior-based (comparison of recent user actions to usual behavior) in the prediction module. At the result, the proposed CIDS will cover all known attack signatures as well as knowledge of new threats. Moreover, using a hybrid detection IDS will increase the accuracy and decrease the false alarm rate significantly [26].

7.2 Dynamic CIDS

The proposed CIDS is a dynamic system through three different perspectives. Firstly, it uses an ensemble system for intrusion detection that combines three well-known deep learning models which have the superiority over traditional machine learning methods in terms of performance. Secondly, it utilizes the double PSO-based algorithm for both feature and hyperparameter selection which makes the architecture of the deep learning models modifies dynamically. This can be explained by the fact that every time the analysis module executes the double PSO-based algorithm, it generates new optimal feature subset from the master database, then it outputs the optimal hyperparameters for each deep learning model that maximizes the accuracy over the reduced master database.

The third perspective is that the response module updates the master database periodically with new predicted “Normal” and “Attack” data records. Accordingly, this makes the master database representative dataset of real-world traffic. Furthermore, in the case of an attack, the response module enforces the analysis module to execute the double PSO-based algorithm again for new feature and hyperparameter selection which makes the deep learning models taking into account the history of all previous predictions in the master database when generating a new prediction for an unseen data record.

7.3 Multithreaded CIDS

Equally important is that the cloud computing generates an enormous amount of data due to a high network access rate. Therefore, cloud-based IDSs should be robust against false positives and noise data. In general, the traditional IDSs are not sufficient for cloud like environments for many reasons such as these IDSs are single threaded and the cloud computing has a massive network traffic that doomed to failure to handle such a large data flow, and due to rich dataset flow [26].

In order to overcome this concern, we designed the proposed CIDS to be a multithreaded IDS, that is, every module operates separately inside an independent thread of the same server. This results in the multithreaded CIDS being able to process large amount of data, reduce the packet loss, and avoid overloading of the VM hosting IDS. As a result, the proposed CIDS will improve the performance over the cloud infrastructure.

For the sake of further improvement, we also added levels of multithreading in our design of CIDS. Whereas the top level contains all the main modules, the bottom level includes the entire functions of each module. Figure 8 shows the mechanism of the proposed multithreaded CIDS. Finally, we believe that such a Parent-Child hierarchy will provide a fast and quick processing.

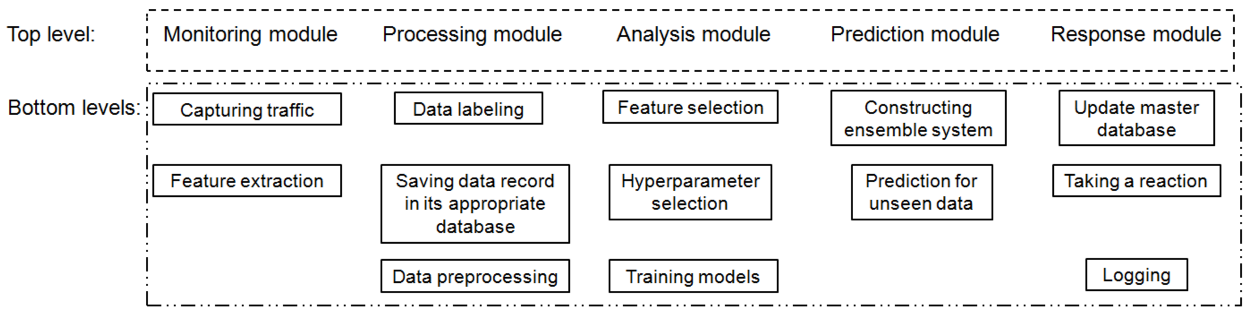

Figure 8: The mechanism of multithreading in the proposed CIDS.

7.4 Transparent CIDSS

The proposed CIDSS acts as an intermediate gate between the user or CSP and the the proposed CIDS. It not only controls the entire CIDS, but it also provides transparency for both the users and CSP. It keeps the users and CSP up-to-date by sending periodic reports to them in order to inform them about any intrusions happened.

7.5 Effective CIDS

In order to evaluate the performance of the proposed CIDS, an empirical experiment on real data is performed. We utilized two well-known IDS datasets, namely, NSL-KDD and CICIDS2017 datasets to test our ensemble system against single DNN, LSTM-RNN, and DBN models. Table 3 presents the results of the empirical experiment. The underlined values in Table 3 refer to the best score of the particular dataset. Based on these findings, our ensemble system outperformed other models in terms of Detection Rate (DR) and False Alarm Rate (FAR) in all datasets.

Table 3: Results of an empirical experiment on NSL-KDD and CICIDS2017 datasets.

Dataset	Optimiztion algorithm	Model	DR (%)	FAR (%)
NSL-KDD	PSO	DNN	92.22	2.4
		LSTM-RNN	95.36	1.66
		DBN	98.8	1.55
		Ensemble	99.3	1.17
	Double PSO	DNN	96.38	0.51
		LSTM-RNN	98.18	0.39
		DBN	99.81	0.23
		Ensemble	**99.9**	**0.16**
CICIDS2017	PSO	DNN	94	1.9
		LSTM-RNN	95.38	1.3
		DBN	96.68	0.8
		Ensemble	98.5	0.5
	Double PSO	DNN	97.58	0.28
		LSTM-RNN	98.68	0.16
		DBN	99.9	0.1
		Ensemble	**99.98**	**0.06**

In order to interpret the results of Table 3 visually, we also plotted the Critical Difference Diagram (CDD) [16]. Figure 9 illustrates the CDD of the tested models on all datasets. Finally, we obtained the Critical Difference (CD) value which is shown as a bar above the figure and equals 3.3166.

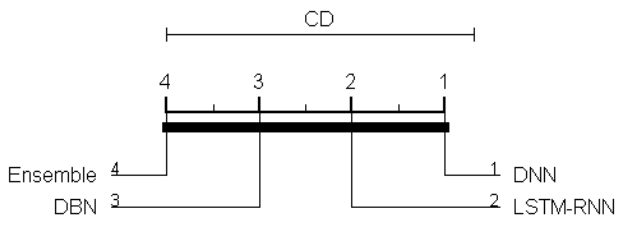

Figure 9: The critical difference diagram of the tested models on all datasets.

7.6 Computational complexity

The core part of the proposed CIDS is the analysis and prediction modules. Regarding the analysis module, a double PSO-based algorithm is applied to select the optimal feature subset as well as the optimal hyperparameters. As mentioned before, the double PSO-based algorithm utilized the FPSBPSO-E algorithm for feature selection which have a complexity of $O(pnlog(n))$ where n is the initial population size and p is the number of iterations. Further, the double PSO-based algorithm exploited a PSO-based algorithm for hyperparameter selection and since the PSO-based hyperparameter selection algorithm computes fitness function for each particle, updates each particles personal best vector, and finally updates the global best vector for the swarm regarding all iterations, the complexity become $O(n^4)$. In order to emphasize the computational overhead, we benefited from the Rosenbrock function to reveal the performance of the FPSBPSO-E algorithm, and we observed that it was more consistent but less sensitive to the choice of hyperparameters whereas the PSO-based hyperparameter selection algorithm produced better results than any other alternative. However, it has a slower convergence on locating the global minimizer. When comparing both approaches, the PSO-based hyperparameter selection algorithm requires almost double the resources and execution time against FPSBPSO-E algorithm [21].

On the other hand, the prediction module depends on a bagging ensemble system of three deep learning models to predict intrusions. We can distinguish between three phases of the bagging ensemble system: training, search of the optimal ensemble, and inference. It is reported that the computational complexity of the training phase is $O(B(N+R))$, where N is the number of classifiers, R is the number of replacements, and B is the number of bags. Meanwhile the computational complexity of the searching phase is $O(N)$, the computational complexity of the inference phase is $O(B)$ [23].

7.7 Limitations

The proposed system might suffer from two potential limitations. Firstly, the Processing module used a signature-based detection procedure in creating the master database. The signature-based detection relies on a set of rules that defined solely by the CSP to determine which records are attack or not. If these rules are not updated enough to classify the attacks correctly, then the master database may have several misclassified records inside and then the process of the analysis module which depends on the master database may be affected significantly. Secondly, the proposed multithreaded CIDS needs a lot of resources that one server cannot afford. Therefore, the process of multithreading can be divided to a cluster of machines in order to benefit from the parallel computation capability.

8 Conclusion

The prevention of intrusion is deemed to be the cornerstone of cloud security. Despite of the excessive work that has been introduced on cloud intrusion detection in the last decade, finding a cloud-based intrusion detection system with effective intrusion detection mechanism is still desirable. In this study, a new integrated CIDS is proposed with full details of its design being explained. It merged five modules which cooperate together to accomplish multiple tasks. The monitoring module acts as an auditing system to capture traffic flows and extract 40 features from them. The processing module is responsible for creating and updating the databases, and to do data preprocessing when needed. In addition to that, the analysis module leverages from a double PSO-based algorithm for feature and hyperparameter selection to train three deep learning models. Then, the prediction module constructs a bagging ensemble system of the trained models to predict intrusions. Finally, the response module takes a reaction upon the decision of the prediction module as well as logs all events happened. Furthermore, a reliable third-party cloud service is proposed to control the performance of CIDS and send reports to the user and CSP. To our knowledge, the design of the proposed CIDS and CIDSS resolves all the limitations and shortcomings of the existing cloud intrusion detection systems in the literature.

Conflict of Interests: The authors declare that there is no conflict of interest regarding the publication of this paper.

Funding: This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] Alguliev R. Abdullaeva F., Illegal access detection in the cloud computing environment, Journal of Information Security, 2014, 5(02), 65.

[2] Alsafi H. M., Abduallah W. M., Pathan A.-S. K., Idps: An integrated intrusion handling model for cloud, arXiv preprint arXiv:1203.3323, 2012.

[3] Ambusaidi M. A., He X., Nanda P., Tan Z., Building an intrusion detection system using a filter-based feature selection algorithm, IEEE transactions on computers, 2016, 65(10), 2986–2998.

[4] Aminanto E. Kim K., Deep learning in intrusion detection system: An overview, 2016 International Research Conference on Engineering and Technology (2016 IRCET), Higher Education Forum, 2016.

[5] Aminanto M. E. Kim K., Deep learning-based feature selection for intrusion detection system in transport layer, Proceedings of the Korea Institutes of Information Security and Cryptology Conference, 2016, 740–743.

[6] Bakshi A. Dujodwala Y. B., Securing cloud from ddos attacks using intrusion detection system in virtual machine, 2010 Second International Conference on Communication Software and Networks, IEEE, 2010, 260–264.

[7] Borji A., Combining heterogeneous classifiers for network intrusion detection, Annual Asian Computing Science Conference, Springer, 2007, 254–260.

[8] Bosin A., Dessì N., Pes B., A service based approach to a new generation of intrusion detection systems, 2008 Sixth European Conference on Web Services, IEEE, 2008, 215–224.

[9] Canadian Institute for Cybersecurity - IDS 2017, http://www.unb.ca/cic/datasets/ids-2017.html, 2017.

[10] Cervante L., Xue B., Zhang M., Shang L., Binary particle swarm optimisation for feature selection: A filter based approach, 2012 IEEE Congress on Evolutionary Computation, IEEE, 2012, 1–8.

[11] Chebrolu S., Abraham A., Thomas J. P., Feature deduction and ensemble design of intrusion detection systems, Computers & security, 2005, 24(4), 295–307.

[12] CICFlowMeter, https://www.unb.ca/cic/research/applications.html#CICFlowMeter, 2017.

[13] CIDDS-Coburg Intrusion Detection Data Sets, https://www.hs-coburg.de/index.php?id=927, 2017.

[14] Dasarathy B. V. Sheela B. V., A composite classifier system design: Concepts and methodology, Proceedings of the IEEE, 1979, 67(5), 708–713.

[15] Dastjerdi A. V., Bakar K. A., Tabatabaei S. G. H., Distributed intrusion detection in clouds using mobile agents, 2009 Third International Conference on Advanced Engineering Computing and Applications in Sciences, IEEE, 2009, 175–180.

[16] Demšar J., Statistical comparisons of classifiers over multiple data sets, Journal of Machine learning research, 2006, 7(Jan), 1–30.

[17] Dhage S. N., Meshram B., Rawat R., Padawe S., Paingaokar M., Misra A., Intrusion detection system in cloud computing environ-

ment, Proceedings of the International Conference & Workshop on Emerging Trends in Technology, 2011, 235–239.

[18] Dzeroski S. Zenko B., Is combining classifiers better than selecting the best one?, ICML, Citeseer, 2002, 2002, 123e30.

[19] Elmasry W., Akbulut A., Zaim A. H., Deep learning approaches for predictive masquerade detection, Security and Communication Networks, 2018, 2018.

[20] Elmasry W., Akbulut A., Zaim A. H., Empirical study on multiclass classification-based network intrusion detection, Computational Intelligence, 2019, 35(4), 919–954.

[21] Elmasry W., Akbulut A., Zaim A. H., Evolving deep learning architectures for network intrusion detection using a double pso metaheuristic, Computer Networks, 2020, 168, 107042.

[22] Elmasry W., Akbulut A., Zaim A. H., Comparative evaluation of different classification techniques for masquerade attack detection, International Journal of Information and Computer Security, 2020, 13(2), 187–209.

[23] Fersini E., Messina V., Pozzi F., Sentiment analysis: Bayesian ensemble learning, Decision Support Systems, 2014, 68.

[24] Folino G. Sabatino P., Ensemble based collaborative and distributed intrusion detection systems: A survey, Journal of Network and Computer Applications, 2016, 66, 1–16.

[25] Gharib A., Sharafaldin I., Lashkari A. H., Ghorbani A. A., An evaluation framework for intrusion detection dataset, 2016 International Conference on Information Science and Security (ICISS), IEEE, 2016, 1–6.

[26] Gul I. Hussain M., Distributed cloud intrusion detection model, International Journal of Advanced Science and Technology, 2011, 34(38), 135.

[27] Hinton G., Deng L., Yu D., Dahl G. E., Mohamed A.-r., Jaitly N., Senior A., Vanhoucke V., Nguyen P., Sainath T. N., et al., Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, IEEE Signal processing magazine, 2012, 29(6), 82–97.

[28] Hinton G. E., Osindero S., Teh Y.-W., A fast learning algorithm for deep belief nets, Neural computation, 2006, 18(7), 1527–1554.

[29] Hochreiter S. Schmidhuber J., Long short-term memory, Neural computation, 1997, 9(8), 1735–1780.

[30] Ji C. Ma S., Combinations of weak classifiers, Advances in Neural Information Processing Systems, 1997, 494–500.

[31] Kadam Y., Security issues in cloud computing a transparent view, International Journal of Computer Science Emerging Technology, 2011, 2(5), 316–322.

[32] Kholidy H. A. Baiardi F., Cids: A framework for intrusion detection in cloud systems, 2012 Ninth International Conference on Information Technology-New Generations, IEEE, 2012, 379–385.

[33] Kholidy H. A., Erradi A., Abdelwahed S., Baiardi F., A hierarchical, autonomous, and forecasting cloud ids, 2013 5th International Conference on Modelling, Identification and Control (ICMIC), IEEE, 2013, 213–220.

[34] Kholidy H. A., Erradi A., Abdelwahed S., Attack prediction models for cloud intrusion detection systems, 2014 2nd International Conference on Artificial Intelligence, Modelling and Simulation, IEEE, 2014, 270–275.

[35] Kim J., Kim J., Thu H. L. T., Kim H., Long short term memory recurrent neural network classifier for intrusion detection, 2016 International Conference on Platform Technology and Service (PlatCon), IEEE, 2016, 1–5.

[36] Lashkari A. H., Draper-Gil G., Mamun M. S. I., Ghorbani A. A., Characterization of tor traffic using time based features., In Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP), 2017, 253–262.

[37] Laureano M., Maziero C., Jamhour E., Protecting host-based intrusion detectors through virtual machines, Computer Networks, 2007, 51(5), 1275–1283.

[38] Lee J.-H., Park M.-W., Eom J.-H., Chung T.-M., Multi-level intrusion detection system and log management in cloud computing, 13th International Conference on Advanced Communication Technology (ICACT2011), IEEE, 2011, 552–555.

[39] Lo C.-C., Huang C.-C., Ku J., A cooperative intrusion detection system framework for cloud computing networks, 2010 39th International Conference on Parallel Processing Workshops, IEEE, 2010, 280–284.

[40] Ludwig S. A., Intrusion detection of multiple attack classes using a deep neural net ensemble, 2017 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2017, 1–7.

[41] Marir N., Wang H., Feng G., Li B., Jia M., Distributed abnormal behavior detection approach based on deep belief network and ensemble svm using spark, IEEE Access, 2018, 6, 59657–59671.

[42] Mazzariello C., Bifulco R., Canonico R., Integrating a network ids into an open source cloud computing environment, 2010 Sixth International Conference on Information Assurance and Security, IEEE, 2010, 265–270.

[43] McHugh J., Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory, ACM Transactions on Information and System Security (TISSEC), 2000, 3(4), 262–294.

[44] Mehmood Y., Shibli M. A., Habiba U., Masood R., Intrusion detection system in cloud computing: Challenges and opportunities, 2013 2nd National Conference on Information Assurance (NCIA), IEEE, 2013, 59–66.

[45] MIT Lincoln Laboratory- DARPA Intrusion Detection Evaluation Data Set, https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-data-set, 1998.

[46] Modi C., Patel D., Borisaniya B., Patel H., Patel A., Rajarajan M., A survey of intrusion detection techniques in cloud, Journal of network and computer applications, 2013, 36(1), 42–57.

[47] Naidoo T., McDonald A., Tapamo J.-R., Feature selection for anomaly–based network intrusion detection using cluster validity indices, 2015.

[48] Natesan P. Balasubramanie P., Multi stage filter using enhanced adaboost for network intrusion detection, International Journal of Network Security & Its Applications, 2012, 4(3), 121.

[49] NSL-KDD Dataset, https://github.com/defcom17/NSL_KDD, 2009.

[50] Patel A., Qassim Q., Shukor Z., Nogueira J., Júnior J., Wills C., Federal P., Autonomic agent-based self-managed intrusion detection and prevention system, Proceedings of the South African Information Security Multi-Conference (SAISMC 2010), 2011, 223–234.

[51] Patel A., Taghavi M., Bakhtiyari K., JúNior J. C., An intrusion detection and prevention system in cloud computing: A systematic review, Journal of network and computer applications, 2013, 36 (1), 25–41.

[52] Ragsdale D. J., Carver C., Humphries J. W., Pooch U. W., Adaptation techniques for intrusion detection and intrusion response systems, Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0, IEEE, 2000, 4, 2344–2349.

[53] Ring M., Wunderlich S., Grüdl D., Landes D., Hotho A., Flow-based benchmark data sets for intrusion detection, Proceedings of the 16th European Conference on Cyber Warfare and Security. ACPI, 2017, 361–369.
[54] Ring M., Wunderlich S., Grüdl D., Landes D., Hotho A., Creation of flow-based data sets for intrusion detection, Journal of Information Warfare, 2017, 16(4), 41–54.
[55] Roschke S., Cheng F., Meinel C., Intrusion detection in the cloud, 2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing, IEEE, 2009, 729–734.
[56] Shaikh R. Sasikumar M., Trust framework for calculating security strength of a cloud service, 2012 International Conference on Communication, Information & Computing Technology (ICCICT), IEEE, 2012, 1–6.
[57] Sharafaldin I., Lashkari A. H., Ghorbani A. A., Toward generating a new intrusion detection dataset and intrusion traffic characterization., ICISSP, 2018, 108–116.
[58] Shelke M. P. K., Sontakke M. S., Gawande A., Intrusion detection system for cloud computing, International Journal of Scientific & Technology Research, 2012, 1(4), 67–71.
[59] Spafford E. H. Zamboni D., Intrusion detection using autonomous agents, Computer networks, 2000, 34(4), 547–570.
[60] Stolfo S. J., Fan W., Lee W., Prodromidis A., Chan P. K., Cost-based modeling for fraud and intrusion detection: Results from the jam project, Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00, IEEE, 2000, 2, 130–144.
[61] Tavallaee M., Bagheri E., Lu W., Ghorbani A. A., A detailed analysis of the kdd cup 99 data set, 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, IEEE, 2009, 1–6.
[62] UCI Machine Learning Repository: KDD CUP 1999 Data Set, https://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data, 1999.
[63] Vani R., Towards efficient intrusion detection using deep learning techniques: a review, Int J Adv Res Comput Commun Eng ISO, 2017, 3297, 2007.
[64] Vieira K., Schulter A., Westphall C., Westphall C., Intrusion detection for grid and cloud computing, It Professional, 2009, 12 (4), 38–43.
[65] Wadi M. Elmasry W., Statistical analysis of wind energy potential using different estimation methods for weibull parameters: a case study, Electrical Engineering, 2021, 1–22.
[66] Wahba Y., ElSalamouny E., ElTaweel G., Improving the performance of multi-class intrusion detection systems using feature reduction, arXiv preprint arXiv:1507.06692, 2015.
[67] Wang Y., Wen J., Wang X., Tao B., Zhou W., A cloud service trust evaluation model based on combining weights and gray correlation analysis, Security and Communication Networks, 2019, 2019.
[68] Yee C. G., Shin W. H., Rao G., An adaptive intrusion detection and prevention (id/ip) framework for web services, 2007 International Conference on Convergence Information Technology (ICCIT 2007), IEEE, 2007, 528–534.
[69] Zarrabi A. Zarrabi A., Internet intrusion detection system service in a cloud, International Journal of Computer Science Issues (IJCSI), 2012, 9(5), 308.
[70] Zhou Z., Liu X., Li P., Shang L., Feature selection method with proportionate fitness based binary particle swarm optimization, Asia-Pacific Conference on Simulated Evolution and Learning, Springer, 2014, 582–592.