Inteligencia Artificial 28(76), 124-148
doi: 10.4114/intartif.vol28iss76pp124-148

INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Distributed two phase intrusion detection system using machine learning techniques and underlying big data storage and processing architecture- HDFS

Abhijit Dnyaneshwar Jadhav[1] , Vidyullatha Pellakuri[2] , Ahire Prashant G. [3], Archana A. Chaugule[4], Harish U. Tiwari[4]

[1] Assistant Professor, Department of Computer Engineering, PCCOE&R, Ravet, Pune
[2]Associate Professor, Department of Computer Science & Engineering, Koneru Lakshamaiah Education Foundation, Guntur, A.P., India
[3]Associate Professor, Department of Computer Science and Engineering, Symbiosis Institute of Technology, Pune, India
[4] Professor, PCCOE&R, Ravet, Pune

Abstract It is crucial for organizations to secure their data in the internet era. The use of Intrusion Detection Systems (IDS) implies this security. Several researchers used various tools and methods to implement various IDS models. However, a few performance concerns that must be resolved are crucial from a security standpoint. The problems pertain to the IDS time efficiency referred as timeliness, accuracy as well as the fault tolerance. The proposed model of intrusion detection has two phases of detection. Every phase uses a different set of machine learning algorithms. Phase I employs Support Vector Machine (SVM) and k nearest neighbor (kNN), whereas Phase II uses Decision Tree and Naïve Bayes. This two phase detection takes care of reducing false positives and false negatives. To compensate the execution time of these four techniques, the big data environment—Hadoop Distributed File System (HDFS)—is utilized as the underlying storage and processing structure. With such arrangement of two phases, the model gives accuracy of 97.29% overall for known and unknown attacks. For known attacks it gives 99.49% and for unknown attacks it gives 96.28% accuracy in detecting intrusion. Also, the time efficiency is measured for training and testing of the model, for training with 10,000 records, it took 0.7 seconds which is very efficient as considered to existing systems. The detailed performance achievements are discussed in results section. Also, because of HDFS, it becomes distributed and fault tolerant intrusion detection system.

Keywords: Accuracy, HDFS, Intrusion Detection System, Machine Learning, Two Phase Intrusion Detection.

1 Introduction

The future of every business organization depends on its data, information, and resources; therefore, the security of these assets is crucial. For client communications and business building, such data is communicated frequently through the network within the organization or outside the organization. It is crucial to identify the sources or the information recipients to whom this information is communicated, whether are genuine and authenticated. If anything which is unknown, suspicious is present in the network, then it should be detected and that too well before the time that it damages business assets. An intrusion detection system, also known as a traffic monitoring system or network activities system, is the one that detects suspicious behavior and sends out notifications when it is found [1] [2]. In order to track authenticated connections and identify any unwanted, unauthorized, or malicious connection requests that could harm the company's data, information, or network, it is vital to keep an eye on every connection request that enters the network from outside the

ISSN: 1137-3601 (print), 1988-3064 (on-line)

organization. An intrusion detection system (IDS) continuously examines incoming network data to identify unauthorized and unlawful network connection requests. IDS are essential security tools for commercial enterprises that wish to keep their data and services safe from outside threats. From 1988 onwards, the world has been subjected to cyber attacks and their aftermath. A Cornell University student named Morris created the first computer worm to be distributed over the internet, damaging almost 6000 computers and resulting in repair bills estimated to be between $10 and $100 million [3]. Subsequently, numerous incidents occur globally, revealing the significance of intrusion detection systems.

Figure 1.1 provides an explanation of the function of IDS within the network.

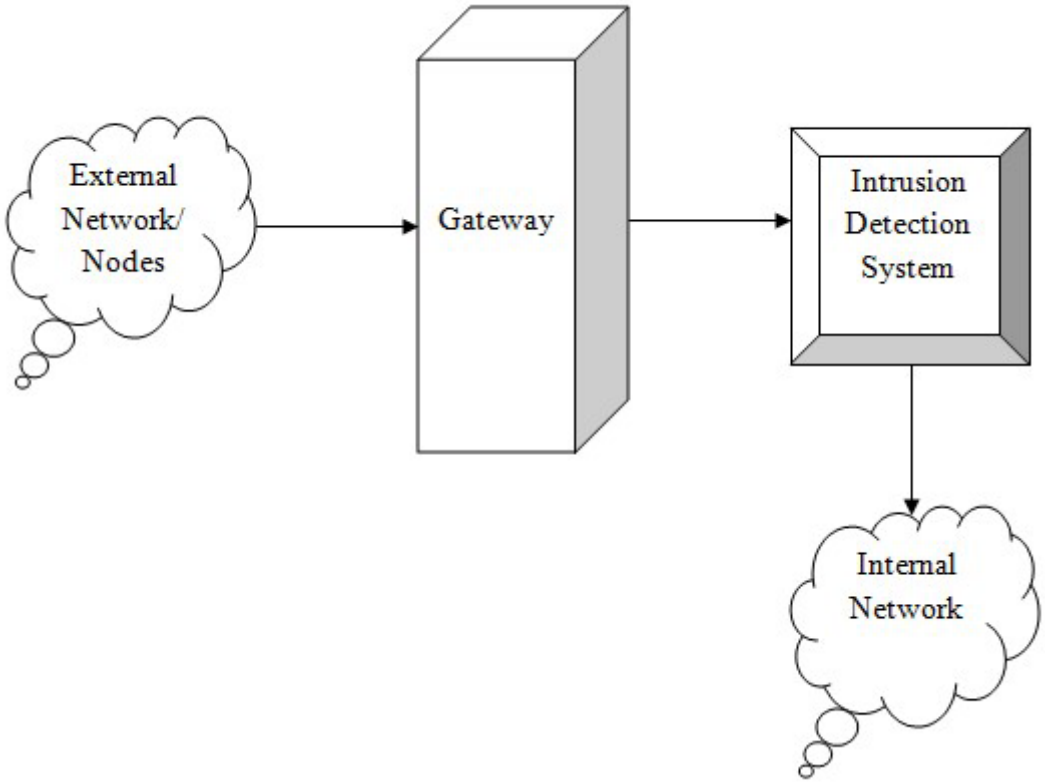


Fig. 1: Intrusion Detection System

Intrusion detection must be installed at numerous points throughout the network due to the many kind of attacks, including host attacks that compromise the system and network attacks that take data [4]. This means that the two types of intrusion detection systems are HIDS, or host-based intrusion detection, and NIDS, or network-based intrusion detection. The NIDS is positioned at the network's edge to monitor incoming data from outside the network [5]. HIDS are host systems that have intrusion detection models installed in order to monitor connection requests and establishments. In order to identify or classify comparable objects into related categories, IDS models analyze all input request patterns that are stored for further use. The investigation's conclusions led to the designation of two types of intrusion detection systems (IDS): Anomaly-based intrusion detection system (AIDS), and signature based detection system (SIDS), also known as signature-based detection system [6]. The phrase "Misuse intrusion detection system" refers to an analytical pattern in intrusion detection wherein the data from incoming connection requests is classed by matching it to previously recorded patterns [7]. These systems can only identify patterns in network traffic that are known to them; they are unable to identify patterns in traffic that are unknown [8]. One of the most annoying challenges for signature-based NIDS is keeping up with a significant volume of incoming traffic when every packet needs to be examined with every signature pattern in the stored database. Consequently, handling all of the traffic takes a long time and reduces the throughput of the system [9]. For a misuse intrusion detection system to function, a signature must be provided for every possible attack that an attacker may attempt within your network. Therefore, in order to maintain the integrity of the signature database in your misuse detection system, regular updates are necessary. Misuse detection presents a challenge since it generates warnings irrespective of the result [10]. For instance, the

SIDS would issue a large amount of alerts for unsuccessful attempts if a window worm attempted to infect a Linux system, making it challenging to deal with. Attack information is therefore environment-specific and heavily dependent on the operating system, its version, and the programs that are executing in the system [11].

The other form of IDS is an anomaly detection system. In this system, the normality behavior line is defined by the IDS, and anything outside of it is identified as anomalous [7] [11]. Since anomaly-based intrusion detection systems have the potential to identify previously unidentified network threats, they have become increasingly popular over time. The benefit of anomaly detection is that it may be used to find new or unknown types of attacks. An alarm is generated by anomaly detection whenever traffic or activity differs from the specified "normal" patterns or activities. This is a drawback. This implies that the task of figuring out why an alarm went off falls on the shoulders of the security administrator. Even though these two IDS system types are more broadly defined in the area of security, efforts are being made to create a hybrid intrusion detection system (IDS) that combines the signature-based Misuse intrusion detection system and the Anomaly intrusion detection system [13] [14]. The reduction of false positives in intrusion detections, which has long been a key problem with IDS system performance optimization, is possible with this hybrid method [15]. In this study, an anomaly detection technique in NIDS is proposed. The focus is on applying machine learning approaches to improve intrusion detection accuracy and timeliness along with fault tolerance. High false positive and high false negative rates, as well as a lengthy detection time for intrusions, are the limits of the work done up until this point.

Researchers employ machine learning techniques in nearly every application domain where historical data is important. Machine learning is a two-step process that involves training the model and evaluating the outcomes to see how much it has learned and to what degree it can accurately classify or forecast data [16]. The quantity and quality of data used in the training phase of machine learning models determines their performance. Hence, machine learning is highly beneficial and produces meaningful outcomes whenever the classifications and predictions rely on past trends or data. The three main categories of machine learning techniques are classification, regression, and clustering [17]. The supervised method for classifying labeled input data is called classification [18]. Classification is a typical supervised learning issue. When we need to predict a categorical type—for example, whether or not a given example falls into a category—we employ it. When it comes to machine learning, the more data used to train the system—assuming that the right data is available—the more accurate the system will be. The three stages of anomaly-based systems—parameterization, training, and detection—are comparable to those of machine learning approaches [19] [20]. Every machine learning model includes a crucial model training step that makes sure the model is learning in order to solve problems automatically through decision-making. By utilizing the commonalities between anomaly-based intrusion detection systems and machine learning models, this solution may be effectively built by creating a model by combining several machine learning techniques. Because the machine learning model for intrusion detection trains and learns from accessible data samples, including attack samples and normal samples, it helps achieve improved accuracy [21]. One benefit of building an intrusion detection system with machine learning techniques is that when the model has been trained on a variety of data samples, it can identify comparable samples or samples with similar behavior with greater accuracy. This study offers a clearer understanding of the comparison between the machine learning and intrusion detection processes.

Here, the Two Phase – intrusion Detection System i.e. TP-IDS architecture's classification technique is being proposed. Identifying a regular or malicious incoming connection request is the initial phase in the process. On the other hand, when the model recognizes an input as being of the attack type, its second phase is utilized to confirm whether or not the results of the first phase are accurate. Different algorithms are available for different classification strategies. In this case, Support Vector Machine (SVM), k nearest neighbor (kNN), Decision Tree, and Naïve Bayes are applied to create the TP-IDS model, which is created in two stages. By adding an accuracy verification step to such a two-phase model, we are expecting to decrease false positives (FP) and false negatives (FN), improving attack detection accuracy. In order to guarantee timeliness in the system, the Hadoop Distributed File System (HDFS) is also utilized as the architecture for parallel processing and data storage for executing the two stages of the intrusion detection model to for timely detection and achieve fault tolerance. The term "TP-IDS," or "Two Phase-Intrusion Detection System," refers to this type of two phase model using HDFS.

Each section of the research article provides a detailed explanation of the research's contribution to the study. The next section elaborates the survey study of IDS. The contributions are explained in sections 3. The experimental design and research implementation outcomes for the specified study objectives are explained in

section 4. The section 5 briefs the conclusion part of the article and extends the dimensions of future scope in this research.

2 Related Work

The survey analysis of intrusion detection system solutions that use machine learning techniques is covered in the section that follows. The survey is divided into four parts.

- **Intrusion Detection System using Support Vector Machine**

This section summarizes the survey of intrusion detection system using support vector machine.

The study on anomalous intrusion detection (IDS) using machine learning techniques by Yihunie and colleagues was published in [22]. The five classification strategies are contrasted with each other as presented in the model. Among the methods used are logistic regression, SGD, random forest, SVM, and sequential model. The models were trained and tested using the NSL KDD. The results demonstrated that random forest produced better results than other algorithms in terms of accuracy. It is important to remember that this accuracy is limited to the detection of known assaults. The most important thing to keep in mind is that the study job does not involve the detection of unknown and unlabeled attack samples. Moreover, neither the system's speed nor the fault tolerance feature of IDS, nor the detection's eternal relevance, are covered in the study. Because of these characteristics, the IDS system has to be improved, and better approaches should be used to expand the model. A machine learning-based intrusion detection system (IDS) model was presented in one of the research articles by Halimaa et al. in [23]. Machine learning classifiers such as Nave Bayes and SVM are used. High accuracy is established by SVM in compared to Nave Bayes. Feature selection is used in model creation to pick important and pertinent characteristics. These accuracy results can only be applied to systems that detect known and labeled assaults; they are not applicable to systems that detect unknown and unlabeled attacks. Another serious problem with the work that has been given is the timeliness. Better results were obtained in [24] when the SVM was used to develop a model for a cloud intrusion detection system (CIDS) utilizing the Correlation based feature selection (COFS). SVM functions well when an efficient feature selection method is used. A comprehensive analysis of the performance of the k nearest neighbor approach for heterogeneous data sets was published by Ali et al. [25]. Teng presented the Collaborative and Adaptive Intrusion Detection (IDS) Model (CAIDM) in [26], which identifies intrusions using SVM, decision trees, and machine learning classification techniques. The KDDCUP99 dataset is used to train and test the CAIDM model. It is found that using SVM and decision tree (DT) together yields better results for intrusion detection than using SVM alone. It's also important to keep in mind that both of these CAIDM approaches are classification techniques, which are useful for identifying attack types that are new but not for prior knowledge assaults. Furthermore, timing is not considered in the model execution. The intrusion detection system (IDS) architecture was created by Agarap and colleagues [27] utilizing neural networks and Gated Recurrent Units (GRU) in conjunction with Support Vector Machines (SVM). SVM is selected if GRU-RNN is used as the output layer in the model as mentioned since it is faster and requires less time complexity. In SVM, only binary classification is employed for testing and training. Multiclassification is the important factor that must be applied in order to produce the outcomes. Moreover, SVM classification produces better results for known attack traffic and ought to be verified against fresh attack traffic.

- **Intrusion Detection System using k nearest neighbor**

This section summarizes the survey of intrusion detection system using kNN.

In the presented research, Li and colleagues reported a kNN-based intrusion detection system (IDS) model and binary classification [28]. There are two components to the model. The model finds anomalous connections in the first phase of binary classification. The next step is kNN, which aids in the detection of unusual and unique input types. When kNN is used, the accuracy of the IDS model is demonstrated to be higher than that of the IDS standalone binary classification.Nevertheless, the accuracy is not verified after the second step, as required by kNN, and the timeliness property value is not investigated in the provided results, which cannot be ignored for the IDS model. The Manhattan distance and Euclidian equations are used to assess the kNN performance. The results show that the Euclidian distance formula does not yield better kNN outputs. Furthermore, the kNN method's performance doesn't significantly alter for heterogeneous nature data sets. The

kNN classification-based intrusion detection (IDS) model was especially proposed for wireless sensor networks (WSNs) by Li et al. in [29]. The type of attack that has been targeted for detection is flooding. The model was found to be effective in distinguishing between various types of flooding attacks using kNN classification. Nevertheless, kNN by itself is inadequate when different attack types or new attack types must be identified. A study using kNN for economic event prediction was presented by Imandoust et al. in [30]. Using the same method as for regression, the value of the property is assigned to the object property by averaging the values of all of its neighbors' properties. Consequently, kNN is used as a useful technique for predicting economic events. The kNN approach is faster and more useful even if you don't know anything about the events beforehand.

- **Intrusion Detection System using naïve bayes**

This section summarizes the survey of intrusion detection system using naïve bayes.

An intrusion detection system (IDS) utilizing the PCA-based Nave Bayes algorithm was introduced by Sharmila et al. in [31]. The results show that the PCA-based nave bayes approach outperforms the conventional nave bayes method in terms of accuracy. This technique also helps to provide insights that are applicable even in cases where data sets include missing values. On the other hand, accuracy decreases and system performance slows down as data volume increases. Therefore, in order to achieve better results, nave bayes might be used in conjunction with other tactics. An intrusion detection system (IDS) that employs the Nave Bayes algorithm was introduced by Panda et al. in [32]. The model produces better results than neural network designs. The model is built in two levels, with as little space as possible between information nodes. The results also demonstrated that the nave bayes approach yields quality results quickly and affordably. The system's drawback is that, in comparison to other systems, it generates more false positives. Therefore, it can be said that Naive Bayes is helpful when combined with other methods to produce efficient outcomes and lower false positives.

- **Intrusion Detection System using decision tree**

This section summarizes the survey of intrusion detection system using decision tree.

Nabila et al. expounded on Intrusion Detection System (IDS) in [33], utilizing the Random Forest approach. The NSL KDD dataset is used to test and train the IDS system. The subset feature selection method is used to eliminate the traits that are pointless and unimportant. When the random forest-based IDS and the J48 classifier technique are compared, it is found that the former produces better results than the latter. The stated accuracy percentages for the identification of known attacks are 99.67%. There is also a notable decrease in false positives (FP) and false negatives (FN). However, its effectiveness in identifying unknown attacks has not been demonstrated, and its execution time is also significantly higher than reasonable. A decision tree-based intrusion detection system was introduced by Kumar et al. in [34]. The outcomes obtained have led to better figures being attained. The decision tree generates better results for recognized input types by building models with pre-existing data. In order to get better results, it should be used in conjunction with other approaches as it is not suitable for use with unknown input types.

- **Intrusion Detection System using other machine learning and deep learning techniques**

This section summarizes the survey of intrusion detection system using other machine learning and deep learning techniques.

Many researchers have attempted the IDS by using other machine learning and deep learning techniques as follows.

Research on anomaly-based intrusion detection with machine learning techniques was done in [35] by Bahlali et al.. The three machine learning techniques that are most frequently used—logistic regression, decision trees, and random forests—as well as the ANN (deep learning) methodology are implemented and compared in this study. USNW-NB15 is the dataset that was used, and it contains issues like imbalanced classes. However, the accuracy obtained by using these classifiers and presented in the results section is satisfactory. Among the employed algorithms, ANN is demonstrated to be the most accurate method for the IDS model. The task's lack of consideration for speed presents a performance problem for the IDS model. Moreover, the results cannot be utilized to properly evaluate the models because the dataset is old and does not reflect current risks. A machine learning-based anomaly-based intrusion detection method was presented by Mazini et al. in [36]. This intrusion detection system was developed with the help of the AdaBoost and

Artificial Bee Colony algorithms. A sequential strategy is used for the execution. The systems testing were conducted using the NSL KDD dataset. When compared to conventional machine learning techniques, the acquired results display higher numbers. Nevertheless, this method uses feature selection, which disregards a number of attributes due to inaccurate feature values, and feature significance is not a requirement. Furthermore, accuracy cannot be extended to hazardous behaviors that are not yet discovered because only known assault samples are classified. The approach is applied in a simulated environment, where the presumptions may not be suitable in an actual situation. Due to the enormous volume of data and high number of connection requests made every second in modern systems, sequential execution is not the best way to achieve timeliness and exhibits poor efficiency or temporal complexity.

In [37], Hu et al developed and put into use a Wi-Fi sensing system for intrusion detection that uses Channel State Information (CSI) as a detection signal at the physical layer of a Wi-Fi network. They employed a deep learning convolutional neural network and the route decomposition technique to increase the sensitivity of a passive intrusion detection system. The computer was able to learn and recognize intrusions without the need to extract numerical properties thanks to CNN. The Channel State Information (CSI) dataset was employed. The average detection accuracy for a single individual in each of the four examples or scenarios they used in their experiment was 98.69%, and for many participants, it was 98.91%. They came to the conclusion that IDSDL can improve system dependability and identify human movements on non-line of sight (NLOS) paths more sensitively than previous techniques.

The urgent problem of cybersecurity in the context of the Internet of Things (IoT), where numerous tiny smart devices send enormous volumes of data over the Internet, is discussed by Alissa et al. in [38]. Understanding that many IoT devices have built-in security vulnerabilities that are made worse by the absence of hardware security support, the study focuses on creating a novel approach to lightweight intrusion detection in IoT settings with limited resources. The suggested method, called Planet Optimization with Deep Convolutional Neural Network for Lightweight Intrusion Detection (PODCNN-LWID), incorporates Planet Optimization (PO) as a hyperparameter tuning procedure and uses a Deep Convolutional Neural Network (DCNN) for intrusion detection.

In [39], Basati and Faghih offer an IoT intrusion detection system that uses an asymmetric parallel auto-encoder (APAE) to identify real-time cyberattacks in the IoT networks. The UNSW-NB15, CICIDS2017, and KDDCup99 datasets are used in this work to train, test, and validate the model. To estimate the identity function for the training data, an APAE is first trained on a dataset in the suggested model. An APAE is first trained on a dataset in the proposed model in order to estimate the identity function for the training data. The final model is then created by combining the transfer layer, encoder, and latent features—the first three elements of the learnt APAE—with a fully connected classifier layer at the end. To ascertain the classifier weights and modify the APAE encoder weights for accurate classification, the final model is then retrained using the training data. When it comes to categorization, the suggested APAE model outperforms the other auto encoder models. In the minority classes, it also does the best categorization. The lightweight NIDS developed in this study is appropriate for Internet of Things networks and devices with constrained memory, computing power, and energy efficiency.

An Autoencoder-based Intrusion Detection System (IDS) for identifying Distributed Denial of Service (DDoS) assaults is presented by Kamalov et al. in a research published in [40]. By highlighting unusual traffic patterns with a larger reconstruction loss, the program detects intrusions. The CSE-CIC-IDS2018 dataset is used in the study's evaluation. When it comes to detecting DDoS assaults, the suggested approach outperforms benchmark unsupervised algorithms.

Based on the literature survey following gaps are identified:

- Need of IDS with enhanced accuracy and reduced false positives and false negatives
- Need of responsive and timely intrusion detection system
- Need of Cost effective and fault tolerant IDS

3 Methodology

The model was created using techniques from machine learning (ML). As shown in the section above, the intrusion detection system model developed using any of the machine learning techniques does not provide all of

the necessary features. The IDS model's performance level is limited when a single approach is used in its design due to the drawbacks of the technology. Consequently, it is imperative to integrate multiple supplementary machine learning techniques to create an intrusion detection system (IDS) model that can provide all the necessary features with optimal performance and without impeding system efficiency. As was also said, the suggested IDS is divided into two phases: Phase I and Phase II.

TP-IDS Phase I

In order to classify unlabeled data more effectively, TP-IDS Phase I consists of two machine learning techniques: Support Vector Machine (SVM) and a similarity-based method called kNN (k closest neighbor).

SVM with RBF kernel

The supervised classification method, or SVM, is helpful for situations involving data classification. On the other hand, the nonlinear radial basis function SVM, or SVM RBF kernel approach, is helpful for classifying non-linearly separable data [41] [42]. Since the data in intrusion detection problems is nonlinear, SVM RBF is a highly helpful tool. The formula used by SVM RBF is as follows:

$$K(X_1, X_2) = \exp\left(-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right) \quad \text{.....eq. (1) [23]}$$

The X_1 and X_2 Euclidean Distances are represented by the expression $||X_1 - X_2||$, and the variance and our hyperparameter for NSL KDD is 'σ'. The "σ" value is crucial for assigning points to the appropriate categories. 'σ' has a variable value depending on the data set [43]. The SVM RBF has limitless capability of classification in nonlinear space because of its exponential nature, which contributes to more accurate and superior outcomes [44]. As a result, it produces an infinite-dimensional hyper plane that produces a highly potent non-linear classifier curve. The input is accepted by SVM RBF as connection request data for categorization. Values for various network factors, such as sender information, transmission type, protocols utilized, security levels employed, server and client information, and any mistakes in transmission, are included in the input. The distance between the input data and the support vectors of the normal class and anomaly class is determined by the RBF kernel formula. When input data is near support vectors relative to other class support vectors, the SVM classifies the data point to the class.

K Nearest Neighbor:

The k closest neighbor method, or kNN algorithm for short, is the most basic machine learning technique. In the kNN approach, k is the number of closest neighbor data points that are analyzed for classifying a new data point in the given categories [45]. The distance between the k nearest data points is calculated using the Euclidian distance, and the new data point is provided as follows:

$$d = \sqrt{((x_1 - x_2)^2 + (y_2 - y_1)^2)} \quad \text{.....eq. (2) [30]}$$

where d represents the Euclidian distance between the two data points, which have coordinates of (x1, y1) and (x2, y2) [46]. The primary benefit of Euclidean distance is that, in the absence of any outliers, the distance between any two items remains unaffected by newly added objects to the analysis.
In TP – IDS phase I, k nearest neighbor serves as an additional technique for classifying the input connection request. The kNN module receives the input connection request data. After receiving the input, the kNN uses data points from the normal class and the anomalous class to determine the input's Euclidian distance. If the input is more similar to k neighbors in the normal class than it is to fewer points in the anomaly class, it is classified to the normal class; if k neighbors are in the anomaly class, it is classed to the anomaly class.

Figure 2 shows the architecture of the system. The architecture is divided into two stages.
The purpose of the first stage is to classify the input as normal or abnormal. The algorithms used in Phase I are SVM and kNN. The incoming input traffic will be routed to Phase II of the TP-IDS and access will be refused if any or all of the SVM and kNN detect it as unusual. In the event that the input is identified by both algorithms as a normal connection request, the request will be approved and access will be provided. In this case, TP-IDS Phase II will not be conducted.

TP-IDS Phase II

TP-IDS Phase II comprises two machine learning (ML) techniques: Decision Tree (DT) and Naïve Bayes (NB), which is a probability-based strategy for more accurate classification of unlabeled data. The identical input sample that was sent to phase I is accepted in this phase.

Decision Tree (CART)

One classification method that works well for several classification issues is the decision tree (CART). Using the specified features of the incoming data, a decision tree technique builds a tree of decision nodes. For a classification task, the interior nodes of the tree represent the decision nodes, and the leaf nodes are the classes [47]. In this case, a binary classification tree called the Classification and Regression Tree, or CART, is employed [48]. The CART classification tree is used as the intrusion detection phase II approach of TP-IDS.

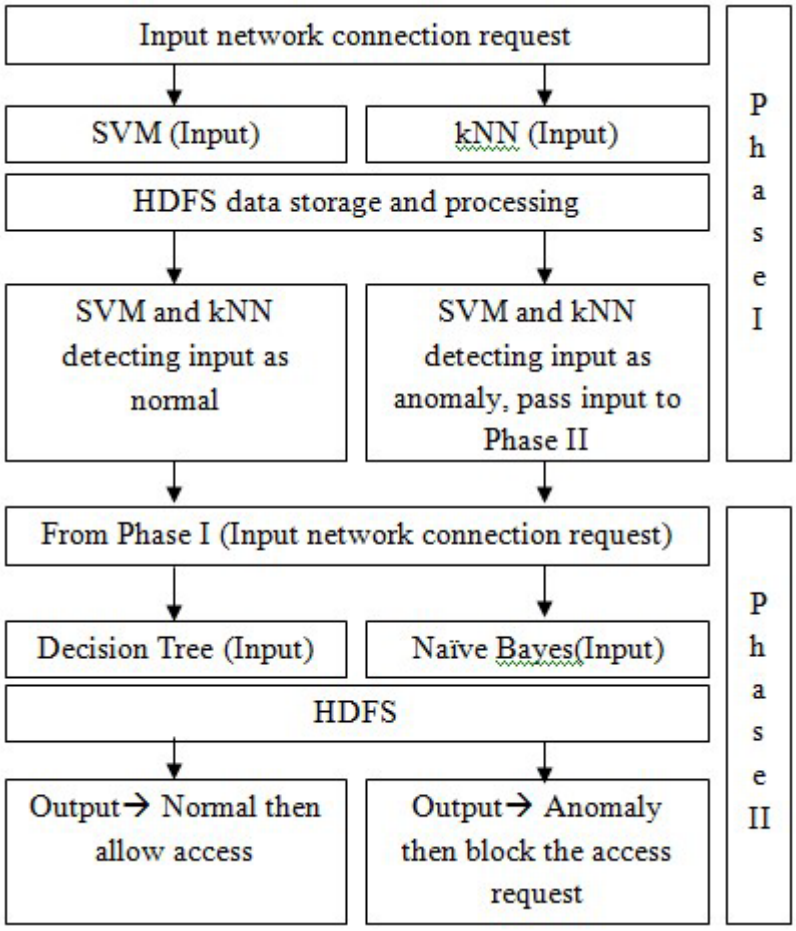


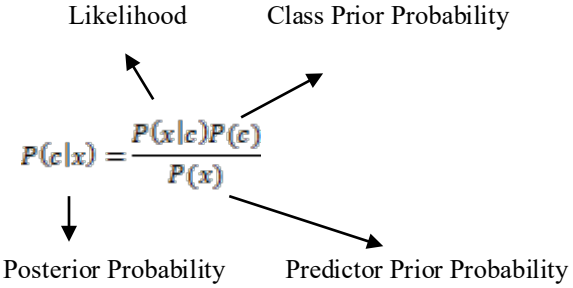
Figure 2 TP-IDS Architecture

The model provides input to CART with a range of properties and values. The properties and their values are extracted using this input. To determine the class of the input, the route of the tree is walked and the values of the input attributes are compared with those of the decision node attributes, starting at the root. The output classes chosen as leaf nodes are determined by comparing the input attribute values with those of the internal decision nodes.

Naïve Bayes

The Naïve Bayes (NB) classifier operates under the assumptions of predictor independence and the Bayes' Theorem [49]. Put simply, a Naive Bayes classifier makes the assumption that each characteristic in a class exists independently of the others. The Naive Bayes model is easy to build and works well with large amounts of data.

Because of its simplicity, Naive Bayes is known to outperform even the most sophisticated classification algorithms. The Bayes theorem allows you to calculate posterior probability P(c|x) by using P(c), P(x), and P(x|c) using P(c), P(x), and P(x|c) [50]. It uses the following formula:



$$P(c|x) = P(x_1|c) \times P(x_2|c) \times P(x_3|c) \times \dots \dots \times P(x_n|c) \times P(c)$$ ……eq. (3) [32]

.Where, The posterior probability of class (c, target) given a predictor is P (c|x) (x, attributes). Prior probability of class is P(c). The likelihood is P (x|c), which is the probability of a predictor given a class. Prior probability of predictor is P(x).

The categorization in TP - IDS phase II is carried out by Naïve Bayes. A classifier based on probability is the Naïve Bayes. The Naïve Bayes algorithm receives the input connection request and uses it to determine the probability values of the input for the output classes—that is, the likelihood that the input will be classified as either normal or anomalous.

In Phase II of TP-IDS, two machine learning techniques are used: decision trees and naive Bayes. In this instance, the TP-IDS validation phase is used. Access to the network will be prohibited if either or both of the decision tree (DT) and naive bayes (NB) detect abnormal input traffic. The ultimate result will be anomalous with attack category designation. In the event that the input connection request is deemed normal by both TP-IDS phase II approaches, access is authorized and the connection request is approved. The Hadoop distributed file system (HDFS) uses the TP-IDS Phase I and Phase II techniques. Because HDFS is distributed, it operates faster and in parallel and is also resilient to faults. By employing Phase II validation, we also increase accuracy and decrease the FPR (False Positive Rate) and FNR (False Negative Rate). If HDFS is used, the two methods used in both phases operate in parallel with each technique's input data processing occurring in parallel. This makes it possible to shorten the processing and detection timeframes of the TP-IDS model. The performance of the TP-intended IDS is aided by this architecture.

HDFS

The underlying data storage and parallel processing architecture for the timely execution of TP-IDS Phase I and Phase II procedures is the Hadoop Distributed File System. While data nodes function as data storage and parallel processing nodes and are able to process the data in parallel, HDFS features a Name node that is master and manages the Meta data consisting of data distribution among various connected data nodes [51]. The parallel data processing nodes for the TP-IDS Phase I and Phase II methods are called HDFS data nodes. Timeliness for TP-IDS is achieved with the use of HDFS in this TP-IDS execution. According to figure 3, the HDFS execution environment for TP-IDS operates as follows. The two phases of the TP-IDS model can be coupled in a pipelined fashion and are meant to be readily deployed separately. The second phase of the pipeline is triggered when an intrusion is discovered using either of the previous phase's methods (kNN or SVM). where the two machine learning methods for intrusion detection—Decision Tree and Naïve Bayes—are used simultaneously. Access is permitted since it is a regular node; nevertheless, connection requests are prohibited if one, both, or neither of these methods detect an intrusion. The method, which consists of the comprehensive working summary steps of TP-IDS employing machine learning and HDFS, is written with clearly specified structures as follows.

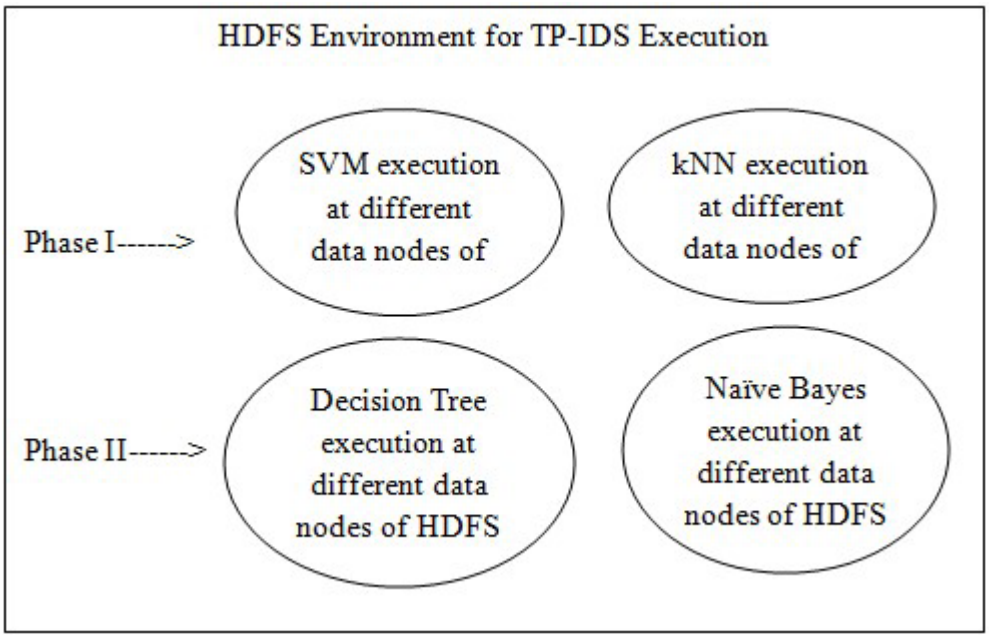


Figure 3 HDFS Execution environment for TP – IDS

The following algorithm along with equation (3) shows the actual implementation flow of the TP-IDS system. The algorithm is implemented in the R programming along with R-Hadoop integration for storing data, and Hadoop is a real time execution environment which speeds up the execution of the TP-IDS model. The parameters used for generating the classification results are as follows:

#	Feature Name	Description	Type	Value Type	Ranges (Between both train and test)
1	Duration	Length of time duration of the connection	Continuous	Integers	0 - 54451
2	Protocol Type	Protocol used in the connection	Categorical	Strings	
3	Service	Destination network service used	Categorical	Strings	
4	Flag	Status of the connection – Normal or Error	Categorical	Strings	
5	Src Bytes	Number of data bytes transferred from source to destination in single connection	Continuous	Integers	0 - 1379963888
6	Dst Bytes	Number of data bytes transferred from destination to source in single connection	Continuous	Integers	0 - 309937401
7	Land	If source and destination IP addresses and port numbers are equal then, this variable takes value 1 else 0	Binary	Integers	{ 0 , 1 }
8	Wrong Fragment	Total number of wrong fragments in this connection	Discrete	Integers	{ 0,1,3 }
9	Urgent	Number of urgent packets in this connection. Urgent packets are packets with the urgent bit	Discrete	Integers	0 - 3

		activated			
10	Hot	Number of “hot“ indicators in the content such as: entering a system directory, creating programs and executing programs	Continuous	Integers	0 - 101
11	Num Failed Logins	Count of failed login attempts	Continuous	Integers	0 - 4
12	Logged In	Login Status : 1 if successfully logged in; 0 otherwise	Binary	Integers	{ 0 , 1 }
13	Num Compromised	Number of "compromised” conditions	Continuous	Integers	0 - 7479
14	Root Shell	1 if root shell is obtained; 0 otherwise	Binary	Integers	{ 0 , 1 }
15	Su Attempted	1 if "su root" command attempted or used; 0 otherwise	Discrete (Dataset contains ‘2’ value)	Integers	0 - 2
16	Num Root	Number of "root" accesses or number of operations performed as a root in the connection	Continuous	Integers	0 - 7468
17	Num File Creations	Number of file creation operations in the connection	Continuous	Integers	0 - 100
18	Num Shells	Number of shell prompts	Continuous	Integers	0 - 2
19	Num Access Files	Number of operations on access control files	Continuous	Integers	0 - 9
20	Num Outbound Cmds	Number of outbound commands in an ftp session	Continuous	Integers	{ 0 }
21	Is Hot Logins	1 if the login belongs to the "hot" list i.e., root or admin; else 0	Binary	Integers	{ 0 , 1 }
22	Is Guest Login	1 if the login is a "guest" login; 0 otherwise	Binary	Integers	{ 0 , 1 }
23	Count	Number of connections to the same destination host as the current connection in the past two seconds	Discrete	Integers	0 - 511
24	Srv Count	Number of connections to the same service (port number) as the current connection in the past two seconds	Discrete	Integers	0 - 511
25	Serror Rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count (23)	Discrete	Floats (hundredths of a decimal)	0 - 1
26	SrvSerror Rate	The percentage of connections	Discrete	Floats (hundredths	0 - 1

		that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in srv_count (24)		of a decimal)	
27	Rerror Rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in count (23)	Discrete	Floats (hundredths of a decimal)	0 - 1
28	SrvRerror Rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in srv_count (24)	Discrete	Floats (hundredths of a decimal)	0 - 1
29	Same Srv Rate	The percentage of connections that were to the same service, among the connections aggregated in count (23)	Discrete	Floats (hundredths of a decimal)	0 - 1
30	Diff Srv Rate	The percentage of connections that were to different services, among the connections aggregated in count (23)	Discrete	Floats (hundredths of a decimal)	0 - 1
31	Srv Diff Host Rate	The percentage of connections that were to different destination machines among the connections aggregated in srv_count (24)	Discrete	Floats (hundredths of a decimal)	0 - 1
32	Dst Host Count	Number of connections having the same destination host IP address	Discrete	Integers	0 - 255
33	Dst Host Srv Count	Number of connections having the same port number	Discrete	Integers	0 - 255
34	Dst Host Same Srv Rate	The percentage of connections that were to different services, among the connections aggregated in dst_host_count (32)	Discrete	Floats (hundredths of a decimal)	0 - 1
35	Dst Host Diff Srv Rate	The percentage of connections that were to different services, among the connections aggregated in dst_host_count (32)	Discrete	Floats (hundredths of a decimal)	0 - 1
36	Dst Host Same Src Port Rate	The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_count (33)	Discrete	Floats (hundredths of a decimal)	0 - 1
37	Dst Host Srv Diff Host Rate	The percentage of connections that were to different destination machines, among the connections aggregated in dst_host_srv_count (33)	Discrete	Floats (hundredths of a decimal)	0 - 1
38	Dst Host Serror Rate	The percentage of connections that have activated the flag (4)	Discrete	Floats (hundredths of a decimal)	0 - 1

		s0, s1, s2 or s3, among the connections aggregated in dst_host_count (32)			
39	Dst Host SrvSerror Rate	The percent of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_srv_count (33)	Discrete	Floats (hundredths of a decimal)	0 - 1
40	Dst Host Rerror Rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_count (32)	Discrete	Floats (hundredths of a decimal)	0 - 1
41	Dst Host SrvRerror Rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_srv_count (33)	Discrete	Floats (hundredths of a decimal)	0 - 1
42	Class	Classification of the traffic input	Categorical	Strings	
43	Difficulty Level	Difficulty level	Discrete	Integers	0 - 21

Table 1: Features /Parameters for TP-IDS from NSL-KDD

TP-IDS Algorithm (Input network traffic)

```
//TP-IDS algorithm is the intrusion detection algorithm which works in two phases.
{
 //Call_Phase_I ()
 //Phase I of TP-IDS is called as first step of TP-IDS
    x=incoming_network_connection_request;
      /* Here x corresponds to the incoming connection request storing values for multiple network
      information attributes */
    If (SVM_RBF_HDFS(x) == attack OR kNN_HDFS(x) == attack)
      /*Calling SVM RBF and kNN as Phase I methods in the HDFS environment for distributed and parallel
      processing */
      {
         //Call_Phase_II ()
         If (DT_HDFS(x) == attack OR NB_HDFS(x) == attack)
         /*Calling Decision Tree and Naïve Bayes as Phase II methods in the HDFS environment for
         distributed and parallel processing */
          {
               Block the connection request.
          }//End of if
         Else
```

{

I/P network traffic is detected as regular user connection and network access is provided.

}//End of else

}//End of if

Else

{

I/P network traffic is detected as regular user connection and network access is provided.

}//End of else

}// End of TP-IDS

The algorithm explains the two phases of the TP-IDS which is the innovative and effective architecture of the intrusion detection system. TP-IDS is proposed in two phases using two machine learning techniques in each phase. In phase I, the SVM with RBF kernel and kNN are called to inspect the incoming connection request. If either of these methods are classifying the incoming connection request as intrusion then phase II techniques decision tree and Naïve Bayes are called. If either of these phase II methods is also classifying the incoming connection request as an intrusion then the access to the network is blocked, otherwise it is classified as a normal connection request and network access is allowed. Also the mathematical model of the system is as follows:

Suppose the x(x1, x2, x3,….,x29) is an input sample with attributes as x1 to x29. Suppose y is an output variable which classifies the input traffic into intrusion or normal category. Consider SVM (x), kNN(x) as phase I functions and DT(x), NB(x) are the phase II functions of the TP-IDS. Then y is,

$$y = \big(SVM(x)\ OR\ kNN(x)\big) AND\ \big(DT(x) OR\ NB(x)\big) \quad \text{…….eq. (4)}$$

The above specified equation is the mathematical model of the TP-IDS system.

4 Results and Discussion

TP-IDS Accuracy

One of the major goals of this study is achieving highly accurate intrusion detection system TP-IDS. The accuracy for the TP-IDS is measured by using confusion matrix and generating values for various parameters like false positives, false negatives, true positives and true negatives. The accuracy values for NSL KDD and CICIDS 2017 datasets is discussed here in this section.

TP-IDS Accuracy for NSL - KDD dataset

As discussed the TP-IDS system model is trained using NSL KDD dataset which has been the most popular dataset. The dataset consists of 43 attributes, where 41 features are the independent features and 42nd feature is the classification type variable along with 43rd rank variable, which tells the rank value of the input type. After applying the correlation based feature selection 29 features are identified as valuable features contributing in identifying the output class value. Table 2 shows the specifics data sample numbers used to train TP-IDS and testing of TP-IDS with NSL KDD:

Dataset	No. of Attack Samples	No. of Normal input samples	Total No. of Samples
NSL KDD Train	58,630	67343	1,25,973
NSL KDD Test	12,833	9711	22,544

Table 2 NSL KDD Dataset samples

The TP-IDS system is trained with NSL KDD Train dataset consisting of 1,25,973 samples comprising of attack samples and normal input samples. Similarly the TP-IDS system is tested using NSL KDD Test dataset with large number of samples 22,544 comprising of both normal behavior samples and attack type samples. Also, to test TP-IDS importantly for known (labeled) and unknown (new) attack samples, the NSL KDD Test dataset is

consisting of sufficient number of samples for both which is shown in table 3:

Dataset	No. of Attack Samples	No. of known attack samples	No. of unknown attack samples
NSL KDD Test	12,833	9,112	3,721

Table 3 NSL KDD Dataset attack samples

Known attack samples are the data samples which are passed during the training of the model and unknown attack samples are new records for TP-IDS, which are not passed during training of the model.
With these samples the TP-IDS is tested and results are separately generated for the known attack inputs and unknown attack inputs as follows:
The results are produced in terms of TPR: true positive rate, TNR: true negative rate, FPR: false positive rate and FNR: false negative rate [52]. These parameter values are evaluated from confusion matrix values using True positives (TP) which are total actual attack samples detected as an attack, True Negatives (TN) which are total actual normal samples detected as normal, False Positives (FP) which are total actual normal samples detected as an attackand False Negatives (FN) which are total actual attack samples detected as normal, obtained for TP-IDS as shown in table 4 and table 5.

	Predicted Attack	Predicted normal
Actual Attack	TP: 9048	FN: 64
Actual Normal	FP: 29	TN: 9682

Table 4 Confusion matrix for TP-IDS for known attack NSL KDD Test samples

	Predicted Attack	Predicted normal
Actual Attack	TP: 3629	FN: 82
Actual Normal	FP: 29	TN: 9682

Table 5 Confusion matrix for TP-IDS for unknown attack NSL KDD Test samples

As per results shown in Table 4 and Table 5 for TP-IDS model, the TPR, TNR, FPR and FNR parameters are calculated using following formulas [53]:

$$TPR = \frac{TP}{TP+FN}$$

$$FPR = \frac{FP}{FP+TN}$$

$$TNR = \frac{TN}{FP+TN}$$

$$FNR = \frac{FN}{TP+FN}$$

The accuracy value of the model is calculated as:

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

Where,
TPR refers to cases that have been accurately classified as positive.
FPR cases that should have been classed as negative but were instead classified as positive.
TNR refers to cases that have been accurately labeled as negatives.
FNR are positive cases that were mistakenly labeled as negative.
Table 5 demonstrates the outcomes for these four parameters for TP-IDS:

	TPR	FNR	TNR	FPR	Accuracy (%)
For known NSL KDD Test samples	99.3	0.7	99.7	0.3	99.53
For unknown NSL KDD Test samples	97.5	2.2	99.7	0.3	99.09

Table 6 Accuracy Parameters for TP-IDS with NSL KDD Test samples

Here Table 6 shows the results for various Accuracy Parameters like FPR, FNR, TPR, TNR and accuracy percentage for TP-IDS using NSL-KDD dataset.

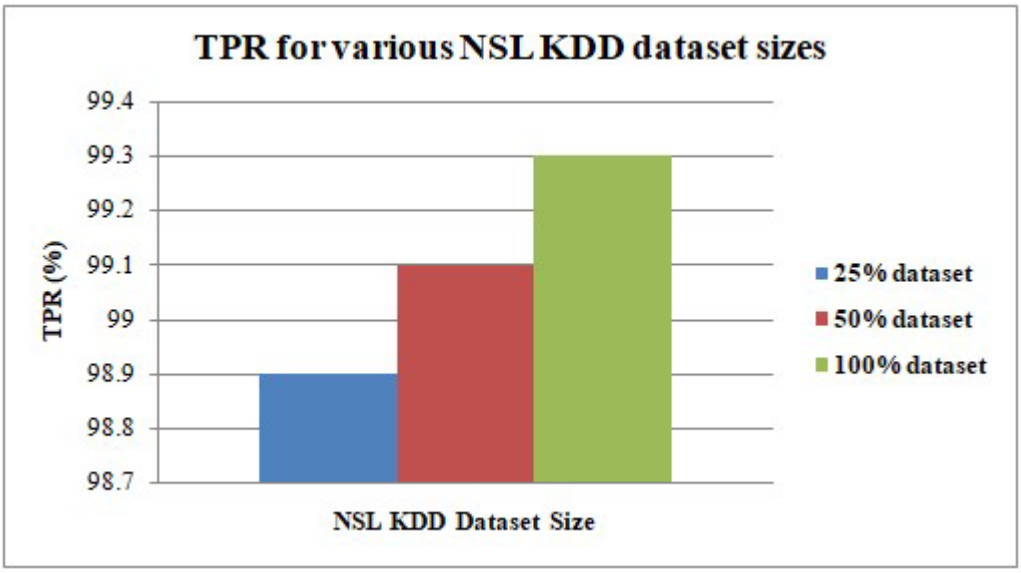


Figure 4 TPR (%) for NSL KDD

Hence, the observation is the FP: false positives and FN: false negative detected with standalone machine learning IDS models are reduced to significant extent and accuracy value is increased with Phase II implementation of TP-IDS.

Figure 4 as above shows the True Positive Rate (TPR) of TP – IDS model for NSL KDD data set.

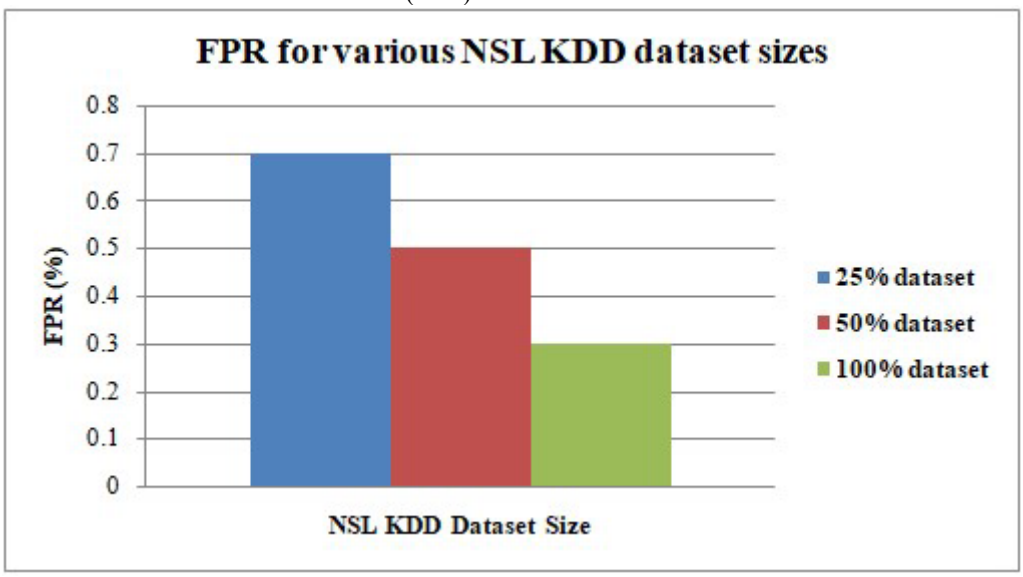


Figure 5 FPR (%) for NSL KDD

Figure 5 shows the False Positive Rate (FPR) of TP – IDS model for NSL KDD data set.

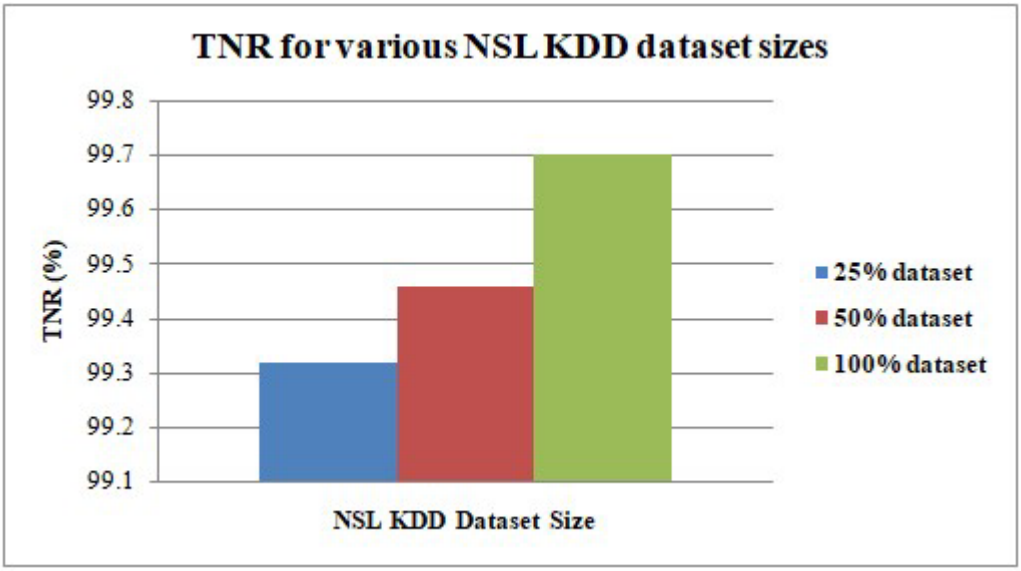


Fig. 6 TNR (%) for NSL KDD

Figure 6 shows the True Negative Rate (TNR) of TP – IDS model for NSL KDD data set.

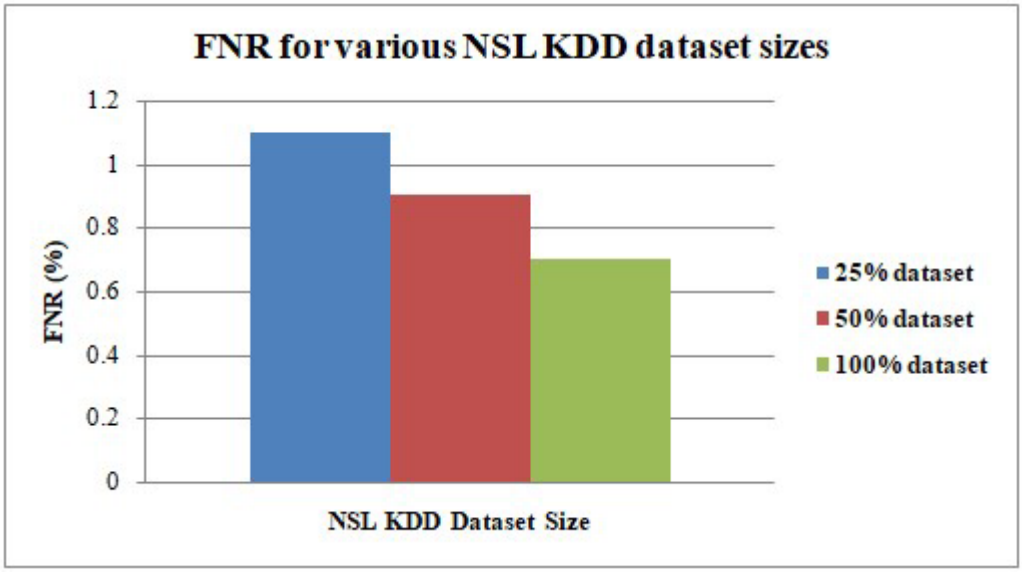


Figure 7 FNR (%) for NSL KDD

Figure 7 shows the False Negative Rate (FNR) of TP – IDS model for NSL KDD data set.

Fig. 8 shows the comparison of the TP-IDS approach with few better existing IDS models. So, the accuracy results for TP-IDS have given better value as compared to existing approaches as depicted with figure 8 for both known (labeled) attack and unknown (new) attack samples in NSL KDD test data set.

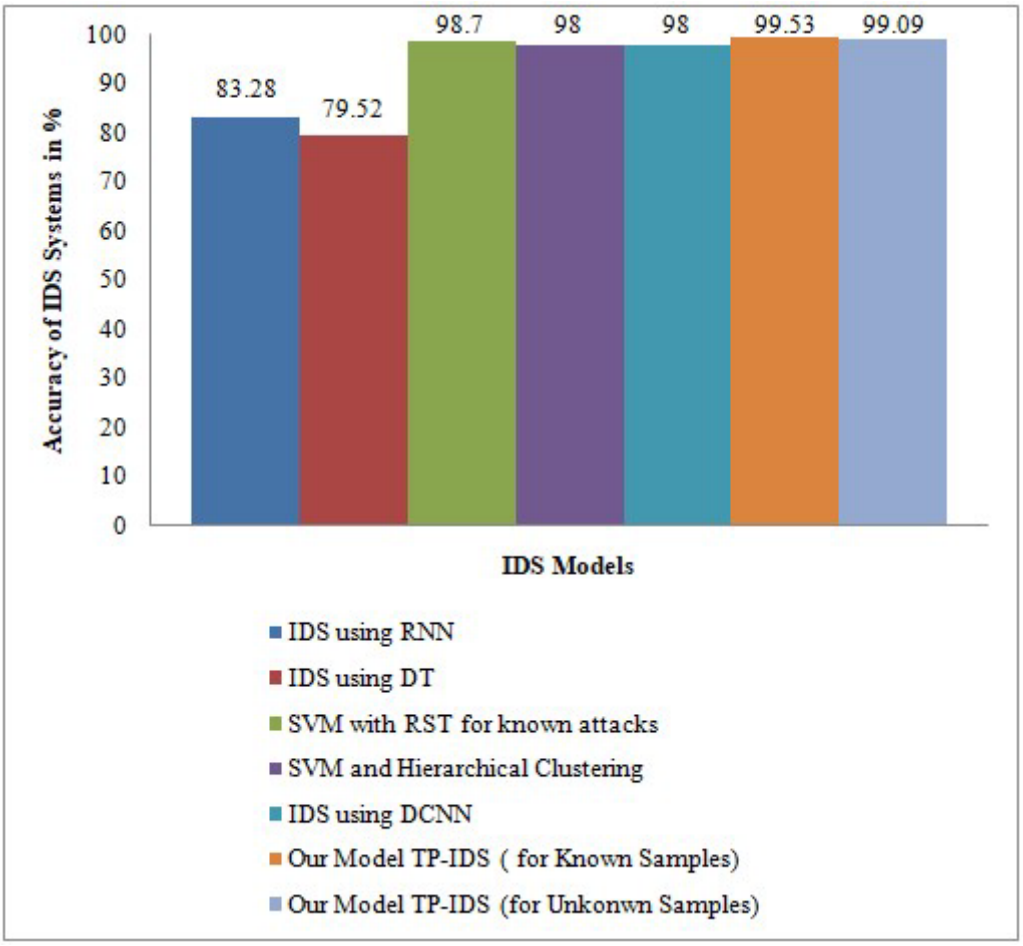


Figure 8 Accuracy results comparison with various IDS models

TP-IDS Accuracy for CICIDS 2017 dataset

The TP-IDS system model is trained and tested using the latest and the effective dataset for researchers in Network Security i.e. CICIDS2017. This dataset consists of real time network data inputs with 83 relevant features. The dataset consists of total 28, 30,540 records without redundancy and without missing values in record samples. This dataset is divided in two parts, with 90% data samples for training of TP-IDS and with 10% data samples to train TP-IDS. Table 7 shows the specifics of the samples used to train and test TP-IDS:

Dataset	No. of Attack Samples	No. of Normal input samples	Total No. of Samples
CICIDS2017 Train	4, 23, 208	21, 24, 278	25, 47, 486
CICIDS2017 Test	48, 245	2, 34, 809	2, 83, 054

Table 7 CICIDS2017 Dataset samples

The TP-IDS system model is trained with CICIDS2017 Train dataset consisting of 25, 47, 486 samples comprising of attack samples and normal input samples. Similarly the TP-IDS system model is tested using CICIDS2017 Test dataset with large number of samples i.e. 2, 83, 054 comprising of both normal type and attack type samples. Also, to test TP-IDS importantly for known and unknown attack samples, the CICIDS2017 Test dataset is consisting of sufficient number of samples for both as described in table 8:

Dataset	No. of Attack Samples	No. of known attack samples	No. of unknown attack samples
CICIDS2017 Test	48, 245	42, 631	5, 614

Table 8 CICIDS2017 Dataset samples

With these samples the TP-IDS is tested and results are separately generated for the known attack inputs and unknown attack inputs as follows:
As discussed in NSL KDD dataset results for TP-IDS, the results are produced for CICIDS2017 in terms of TPR, TNR, FPR and FNR. For this, the confusion matrix for TP-IDS as displayed in table 9 and table 10.

	Predicted Attack	Predicted normal
Actual Attack	TP: 42, 115	FN: 516
Actual Normal	FP: 1, 855	TN: 2, 32, 954

Table 9 Confusion matrix for TP-IDS for known attack CICIDS2017 Test samples

	Predicted Attack	Predicted normal
Actual Attack	TP: 5, 361	FN: 253
Actual Normal	FP: 1, 855	TN: 2, 32, 954

Table 10 Confusion matrix for TP-IDS for unknown attack CICIDS2017 Test samples

	TPR	FNR	TNR	FPR	Accuracy (%)
For known CICIDS2017 Test samples	98.79	1.21	99.21	0.79	99.0
For unknown CICIDS2017 Test samples	95.49	4.5	99.21	0.79	97.35

Table 11 Accuracy Parameters for TP-IDS with CICIDS2017 Test samples

Table 11 shows the overall accuracy of TPIDS model for CICIDS2017 dataset.
After testing the TP-IDS system model with the latest CICIDS2017 dataset, it is discovered that TP-IDS is very useful as provides high accuracy value for both known and unknown attack samples. So, the accuracy results for TP-IDS have given better value as for both the datasets NSL KDD and CICIDS2017. Hence, it is proved that, TP-IDS is very useful and a model for detecting intrusions that is effective by securing the organizational networks.
Also, the ROC curve is generated based on the predictions of output variable and the actual values available in the data records. Fig. 9 shows the ROC curve for TP-IDS model giving significant importance of the model accuracy.

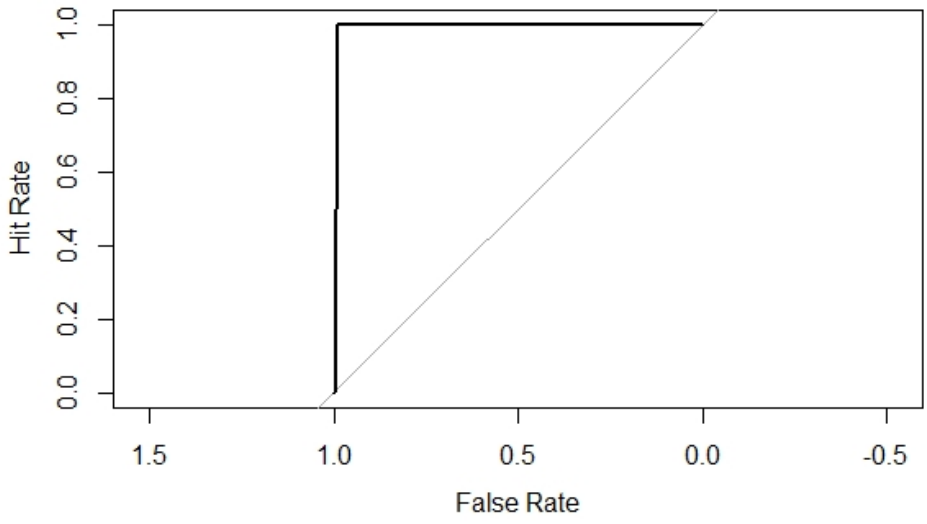


Figure 9 ROC curve for TP-IDS Model

As shown in fig. 9, the area under curve (AUC) for TP-IDS is 0.9949, which is the better values suggesting better accuracy rates for the TP-IDS model. But, still the accuracy with unknown attack samples can be improved and this is one of the challenges that are faced while implementing the TP-IDS architecture. The results generated are considering the R-Hadoop integration, and community hardware systems. If the machine capability is scaled up, then it might increase the overall performance of the TP-IDS.

TP-IDS Timeliness

To speed up the parallel execution of the TP-IDS Phase I and TP-IDS Phase II techniques, the underlying data storage architecture used in HDFS i.e. Hadoop distributed file system.

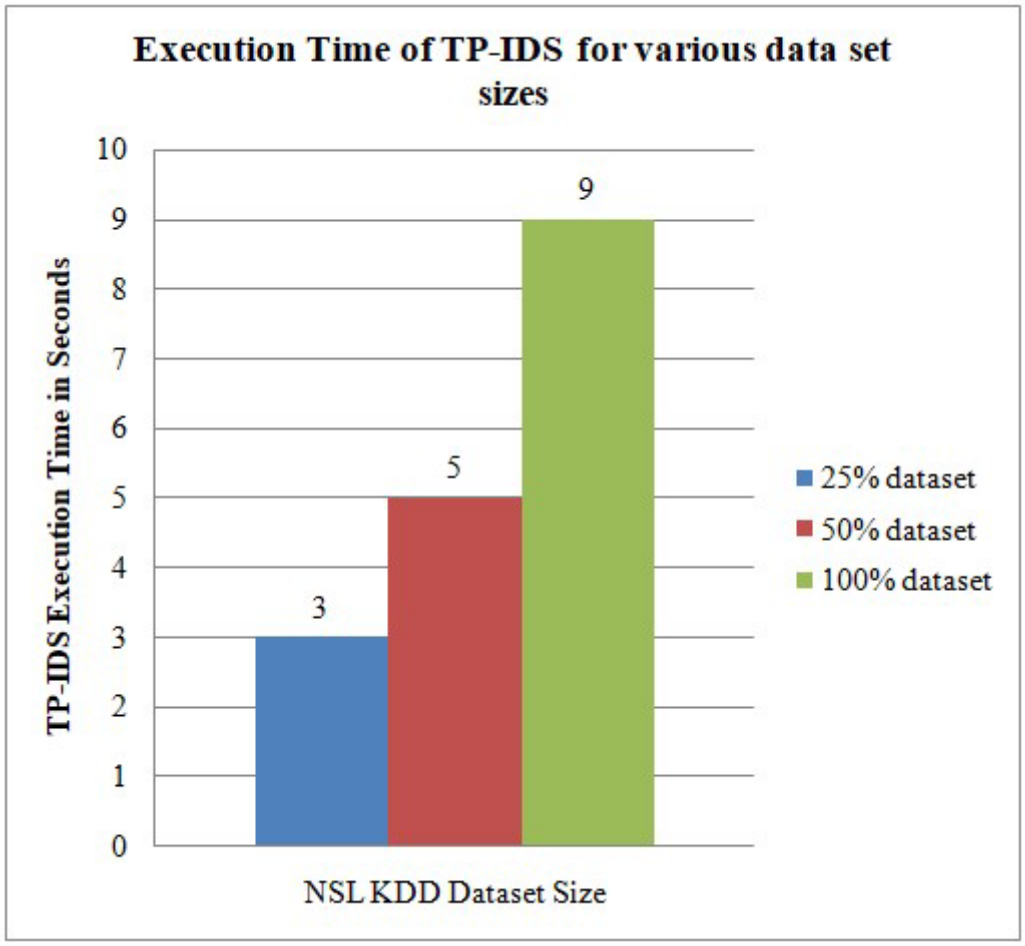


Figure 10 Execution Time for TP-IDS Model for various data sample size of NSL KDD

HDFS is a data storage distributed system and parallel processing file system, which scales the performance of any system to amazing levels. HDFS has aided in greatly increasing the pace of data processing and has produced good results in TP-IDS for classifying input data. The output classifications of the input samples are also created in a timely manner, as predicted, due to the utilization of the Hadoop distributed file system. Even with such a large number of records in the training and testing datasets, in the TP-IDS system model, it provides parallel and quicker data processing. This has helped us in achieving the timeliness in classification of the data. Hence, TP-IDS helps in detecting attacks before any damage to system is done, which is one of the excellent feature of this TP-IDS system model. The fig. 10 and fig. 11 shows the timeliness achievement of TP-IDS.

As displayed in fig. 10, the TP-IDS execution time is tested for different data samples size of NSL KDD dataset samples. The graph is showing the faster data processing ability of TP-IDS with the use of HDFS. Also, as HDFS is distributed file system enabling number of TP-IDS server nodes and each server node of TP-IDS is connected to number of data processing TP-IDS nodes, a distributed TP-IDS model, which makes it a fault tolerant system.

Hence, another advantage we get with the HDFS is fault tolerant intrusion detection system, which ensures that, system will be available in spite of failure of few server nodes.
Fig. 11 shows the TP-IDS execution time comparison with existing IDS models.

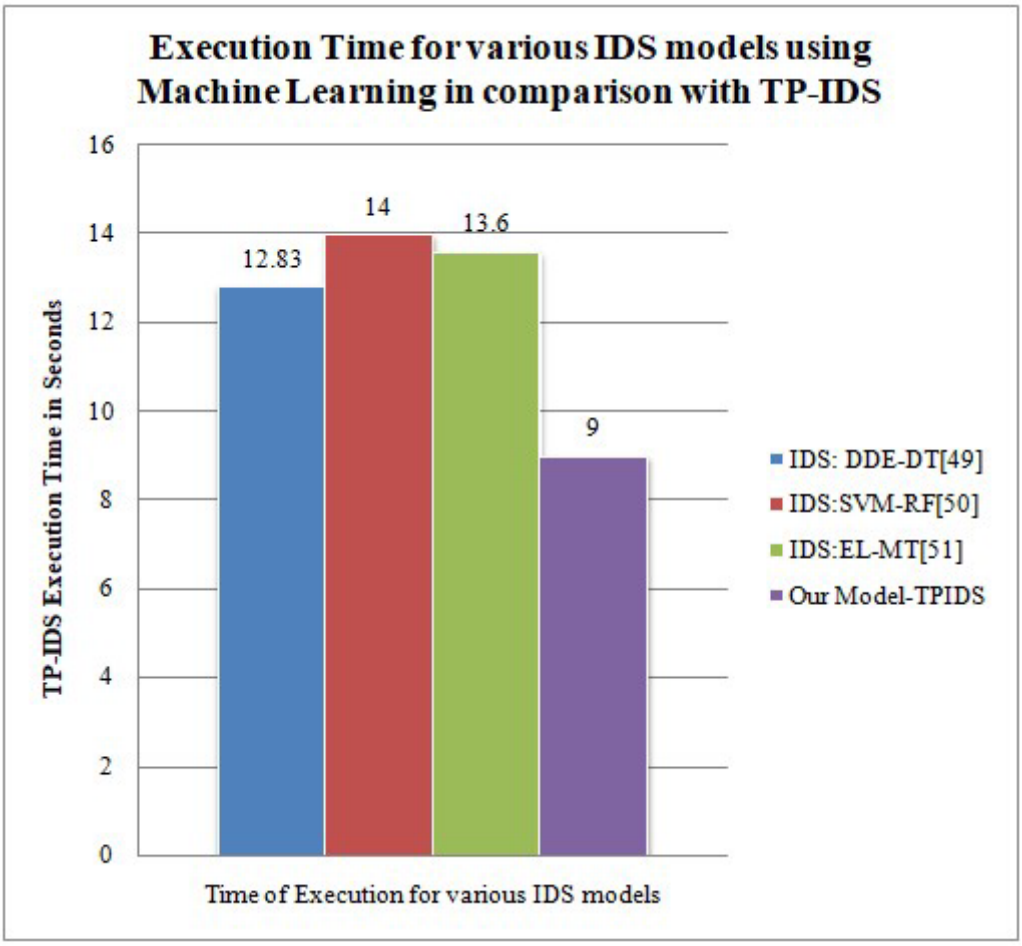


Figure 11TP-IDS execution time comparison with existing IDS models

TP-IDS Evaluation for Other Parameters

TP-IDS is evaluated with respect to other parameters as follows apart from accuracy and timeliness.

Cumulative False Alarm Rate (CFAR):
The weighted average of False Positive and False Negative ratios.

$$CFAR = \frac{FPR + FNR}{2} = \frac{0.3+0.7+2.2+0.3}{4} = 0.875$$

Depth of System's Detection Capability (DSDC):
It is defined as the number of attack signature patterns and/or behavior models known to it. In NSL KDD the data of various 39 attacks is present.
Hence,
DSDC = 39.

Reliability of Attack Detection
Reliability is the false positives to total alarms raised ratio.

$$\text{Reliability} = \frac{FP}{\text{Total Alarms Raised}} = \frac{29}{22544} = 0.00128$$

Possibility of Attack
Possibility is false negatives to true negatives ratio.
Possibility = $\frac{FN}{TN} = \frac{64}{9682} = 0.0066$

User Friendliness:
The ability of IDS to configure according to user's environment. The TP-IDS is flexible enough and can be deployed in any network environment for intrusion detection. Hence, it has complete user friendliness.

Compromise Analysis:

It is the ability to report the extent of damage and compromise due to intrusions. TP-IDS has high detection rate which makes it less prone to damages due to intrusions. Also, as it is using the Hadoop Distributed File System architecture, it's functioning is not dependent on any centralize node which makes TP-IDS as fault tolerant system.

Considering the limitations of TP-IDS, is use of four different machine learning techniques which are all supervised learning techniques, which causes better results for known attack. It needs to be cheeked, if this system performs better in case of severe unknown attack situation or not. This is the limitation and one of the serious challenges that are experienced by us.

Applicability of TP-IDS:

The distributed TP-IDS is basically to secure the systems and network at large scale such as cloud computing environments. In cloud environments, intrusion detection systems (IDS) are essential for safeguarding cloud infrastructure and making sure that private information is shielded from assaults, breaches, and unwanted access. As cloud computing services like Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) become more popular, it is essential to keep robust security measures in place to safeguard cloud resources and the data they contain.

The TP-IDS will help in monitoring and detecting the threats in cloud environment. It is able to continually scan cloud network traffic for questionable activity including brute-force login attempts, spoofing, and DDoS assaults. The IDS has the ability to automatically block malicious traffic and notify administrators when an anomaly is found. Additionally, it detects malicious traffic patterns in real time, which aids cloud service providers in minimizing potential vulnerabilities and blocking harmful access attempts.

The TP-IDS can also help to protect cloud based virtual machines. Cloud virtual machines are susceptible to zero-day vulnerabilities, rootkits, and malware. TP- IDS are able to keep an eye on virtual machines' operating and file systems for any unusual activity that could point to a security breach. It will also help to ensure the data integrity. Data integrity is a key problem in cloud systems, particularly when sensitive data is kept in large quantities across dispersed resources. It can keep an eye on access trends and stop malevolent individuals from altering or destroying data kept on cloud servers. If an unauthorized entity accesses or transfers sensitive data, for instance, it can notify administrators of the unusual data flow trend.

5 Conclusion and Future Work

In the presented research work, it is investigated that, a strong, cost effective, secure and faster intrusion detection (IDS) model is needed. The innovative two phase (TP) -intrusion detection system (IDS) i.e. TP - IDS, made up of two phases, TP - IDS Phase I and TP - IDS Phase II, where TP – IDS Phase I consists of SVM and kNN, TP – IDS Phase II consists of Decision Tree and Naïve Bayes, is designed and implemented to reduce the false positives and false negatives of the intrusion detection system. Also, considering the timeliness of the IDS, the distributed file system HDFS is also added in the design and implementation to speed up the IDS model execution and achieve the timeliness. The R Hadoop environment is used to implement the TP-IDS model and results are generated. The accuracy of the model is observed up to 99.49% approx., which is generated by ROC AUC also. This shows the TP-IDS model has enhanced accuracy value if compared with the other IDS approaches. The execution speed of the TP - IDS system is observed up to 9 seconds for training the TP-IDS using NSL KDD training dataset samples, which is far better and very efficient comparing the performance of existing IDS models. So, it is also justified that, HDFS which is distributed data storage and parallel processing architecture helps in increasing the speed of the TP-IDS model. The distributed nature of the HDFS makes TP-IDS nodes as distributed IDS system with number master nodes and removes the dependency on the centralized

IDS server node, making it fault tolerant system. Hence, finally it is observed that, the TP-IDS which is two phase intrusion detection (IDS) model using machine learning techniques and HDFS file system is innovative implementation with enhanced accuracy and timely fault tolerant system which is desired to protect organizations network from the outsiders and intruders. The future work of the research specifies that, more phases or unsupervised and semi supervised techniques, deep learning techniques can be employed to increase the IDS model's performance for the unknown data samples of the network connection requests. Also, the model can be applied in the realistic environment with different phase configurations and techniques of machine learning to avoid any biased results as an effect of assumptions of implementation environment and predefined data set.

Acknowledgement

We are thankful to PCCOER, Ravet Team and KLEF, Guntur Team for support and motivation.

References

1. Talukder, M.A., Islam, M.M., Uddin, M.A. et al. Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction. J Big Data 11, 33 (2024). https://doi.org/10.1186/s40537-024-00886-w.
2. Marwala T. Cybersecurity in politics. In: Artificial intelligence, game theory and mechanism design in politics. Springer; 2023. p 135–155.
3. Nguyen H, Lim Y, Seo M, et al. Strengthening information security through zero trust architecture: a case study in South Korea. In: International conference on intelligent systems and data science, Springer;2023 pp 63–77.
4. Uddin, Mueen& Abdul Rahman, Azizah& Uddin, Naeem&Memon, Jamshed&Kazi, Suhail. (2013). Signature-based Multi-Layer Distributed Intrusion Detection System using Mobile Agents. International Journal of Network Security. 15. 79-87.
5. Chun Guo, Yuan Ping, Nian Liu and Shou-Shan Luo, A twolevel hybrid approach for intrusion detection, Neurocomputing,http://dx.doi.org/10.1016/j.neucom.2016.06.021
6. Wun-Hwa Chen, Sheng-Hsun Hsu, Hwang-Pin Shen, Application of SVM and ANN for intrusion detection, Computers & Operations Research, Volume 32, Issue 10, 2005, Pages 2617-2634, ISSN 0305-0548,https://doi.org/10.1016/j.cor.2004.03.019.
7. Nabila Farnaaz, M.A. Jabbar, Random Forest Modeling for Network Intrusion Detection System, Procedia Computer Science, Volume 89, 2016, Pages 213-217, ISSN 1877-0509,https://doi.org/10.1016/j.procs.2016.06.047.
8. Sahani, Roma &Shatabdinalini, & Rout, Chinmayee&Badajena, J. & Jena, Ajay & Das, Himansu. (2018). Classification of Intrusion Detection Using Data Mining Techniques. 10.1007/978-981-10-7871-2_72.
9. Rai, Kajal& Devi, Mandalika& Professor, Devi &Guleria, Ajay. (2016). Decision Tree Based Algorithm for Intrusion Detection. International Journal of Advanced Networking and Applications. 07. 2828-2834.
10. G. Karatas, O. Demir and O. K. Sahingoz, "Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset," in IEEE Access, vol. 8, pp. 32150-32162, 2020, doi: 10.1109/ACCESS.2020.2973219.
11. R. Kumari, Sheetanshu, M. K. Singh, R. Jha and N. K. Singh, "Anomaly detection in network traffic using K-mean clustering," 2016 3rd International Conference on Recent Advances in Information Technology (RAIT), 2016, pp. 387-393, doi: 10.1109/RAIT.2016.7507933.
12. Z. Li, Y. Li and L. Xu, "Anomaly Intrusion Detection Method Based on K-Means Clustering Algorithm with Particle Swarm Optimization," 2011 International Conference of Information Technology, Computer Engineering and Management Sciences, 2011, pp. 157-161, doi: 10.1109/ICM.2011.184.
13. Munther, A. & Othman, RozmieRazif&Abualhaj, Mosleh& Anbar, Mohammed &Yaakob, Shahrul. (2016). A Preliminary Performance Evaluation of K-means, KNN and EM Unsupervised Machine Learning Methods for Network Flow Classification. International Journal of Electrical and Computer Engineering (IJECE). 6. 778-784. 10.11591/ijece.v6i1.8909.
14. H. Yao, D. Fu, P. Zhang, M. Li and Y. Liu, "MSML: A Novel Multilevel Semi-Supervised Machine Learning Framework for Intrusion Detection System," in IEEE Internet of Things Journal, vol. 6, no. 2, pp. 1949-1959, April 2019, doi: 10.1109/JIOT.2018.2873125.
15. X. Gao, C. Shan, C. Hu, Z. Niu and Z. Liu, "An Adaptive Ensemble Machine Learning Model for Intrusion Detection," in IEEE Access, vol. 7, pp. 82512-82521, 2019, doi:

10.1109/ACCESS.2019.2923640.
16. Abdulhammed R, Musafer H, Alessa A, Faezipour M, Abuzneid A. Features Dimensionality Reduction Approaches for Machine Learning Based Network Intrusion Detection. Electronics. 2019; 8(3):322. https://doi.org/10.3390/electronics8030322
17. U. S. Musa, M. Chhabra, A. Ali and M. Kaur, "Intrusion Detection System using Machine Learning Techniques: A Review," 2020 International Conference on Smart Electronics and Communication (ICOSEC), 2020, pp. 149-155, doi: 10.1109/ICOSEC49089.2020.9215333.
18. Vipin, Das &Vijaya, Pathak &Sattvik, Sharma &Sreevathsan, &MVVNS.Srikanth, & T, Gireesh. (2010). Network Intrusion Detection System Based On Machine Learning Algorithms. International Journal of Computer Science & Information Technology. 2. 10.5121/ijcsit.2010.2613.
19. P V R N S S V Sai Leela, BankapalliJyothi, P. I. priyadarsini. (2021). Towards Intelligent Machine Learning Models for Intrusion Detection System. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(5), 643â€“655. doi:10.17762/turcomat.v12i5.1062
20. Waheed, Sajjad&Istiaque, Shah & Khan, Asif. (2020). Smart Intrusion Detection System Comprised of Machine Learning and Deep Learning. International Journal of Scientific and Engineering Research. 5. 1. 10.24018/ejers.2020.5.10.2128.
21. M. Banadaki, Y. (2020). Evaluating the performance of machine learning algorithms for network intrusion detection systems in the internet of things infrastructure. Journal of Advanced Computer Science & Technology, 9(1), 14-20. doi:http://dx.doi.org/10.14419/jacst.v9i1.30992
22. F. Yihunie, E. Abdelfattah and A. Regmi, "Applying Machine Learning to Anomaly-Based Intrusion Detection Systems," 2019 IEEE Long Island Systems, Applications and Technology Conference (LISAT), Farmingdale, NY, USA, 2019, pp. 1-5, doi: 10.1109/LISAT.2019.8817340.
23. A. Halimaa A. and K. Sundarakantham, "Machine Learning Based Intrusion Detection System," 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 2019, pp. 916-920, doi: 10.1109/ICOEI.2019.8862784.
24. W. Wang, X. Du and N. Wang, "Building a Cloud IDS Using an Efficient Feature Selection Method and SVM," in IEEE Access, vol. 7, pp. 1345-1354, 2019, doi: 10.1109/ACCESS.2018.2883142.
25. Ali, N., Neagu, D. & Trundle, P. Evaluation of k-nearest neighbour classifier performance for heterogeneous data sets. SN Appl. Sci. 1, 1559 (2019). https://doi.org/10.1007/s42452-019-1356-9.
26. Mazini, M., Shirazi, B., Mahdavi, I., Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms, Journal of King Saud University – Computer and Information Sciences (2018), doi: https://doi.org/10.1016/j.jksuci.2018.03.011.
27. Agarap, Abien Fred. (2017). A Neural Network Architecture Combining Gated Recurrent Unit (GRU) and Support Vector Machine (SVM) for Intrusion Detection in Network Traffic Data. 10.1145/3195106.3195117.
28. L. Li, Y. Yu, S. Bai, Y. Hou and X. Chen, "An Effective Two-Step Intrusion Detection Approach Based on Binary Classification and k -NN," in IEEE Access, vol. 6, pp. 12060-12073, 2018, doi: 10.1109/ACCESS.2017.2787719.
29. Wenchao Li, Ping Yi, Yue Wu, Li Pan, Jianhua Li, "A New Intrusion Detection System Based on KNN Classification Algorithm in Wireless Sensor Network", Journal of Electrical and Computer Engineering, vol. 2014, Article ID 240217, 8 pages, 2014. https://doi.org/10.1155/2014/240217.
30. Imandoust, S.B. &Bolandraftar, Mohammad. (2013). Application of K-nearest neighbor (KNN) approach for predicting economic events theoretical background. Int J Eng Res Appl. 3. 605-610.
31. B. S. Sharmila and R. Nagapadma, "Intrusion Detection System using Naive Bayes algorithm," 2019 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), Bangalore, India, 2019, pp. 1-4, doi: 10.1109/WIECON-ECE48653.2019.9019921.
32. Panda, Mrutyunjaya&Patra, Manas. (2007). Network intrusion detection using naive bayes. 7.
33. Farnaaz, Nabila &Akhil, Jabbar. (2016). Random Forest Modeling for Network Intrusion Detection System. Procedia Computer Science. 89. 213-217. 10.1016/j.procs.2016.06.047.
34. M. Kumar, M. Hanumanthappa and T. V. S. Kumar, "Intrusion Detection System using decision tree algorithm," 2012 IEEE 14th International Conference on Communication Technology, Chengdu, China, 2012, pp. 629-634, doi: 10.1109/ICCT.2012.6511281.
35. Bahlali, Ahmed Ramzi. (2019). Anomaly-Based Network Intrusion Detection System: A Machine Learning Approach. 10.13140/RG.2.2.29553.84325.
36. Mazini, M., Shirazi, B., Mahdavi, I., Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms, Journal of King Saud University – Computer and

Information Sciences (2018), doi: https://doi.org/10.1016/j.jksuci.2018.03.011.
37. Hu, Y., Bai, F., Yang, X., Liu, Y.: Idsdl: a sensitive intrusion detection system based on deep learning. EURASIP Journal on Wireless Communications and Networking 2021(1), 95 (2021) https://doi.org/10.1186/s13638-021-01900-y
38. A. Alissa, K., S. Alrayes, F., Tarmissi, K., Yafoz, A., Alsini, R., Alghushairy, O., Othman, M., Motwakel, A.: Planet optimization with deep convolutional neural network for lightweight intrusion detection in resource-constrained iot networks. Applied Sciences 12(17) (2022) https://doi.org/10.3390/app12178676
39. Basati, A., Faghih, M.M.: Apae: an iot intrusion detection system using asymmetric parallel auto-encoder. Neural Computing and Applications 35(7),4813–4833 (2023) https://doi.org/10.1007/s00521-021-06011-9
40. Kamalov, F., Zgheib, R., Leung, H.H., Al-Gindy, A., Moussa, S.: Autoencoderbased intrusion detection system. In: 2021 International Conference on Engineering and Emerging Technologies (ICEET), pp. 1–5 (2021). https://doi.org/10. 1109/ICEET53442.2021.9659562
41. Mulay, Snehal&Devale, P.R. &Garje, Goraksh. (2010). Intrusion Detection System Using Support Vector Machine and Decision Tree. International Journal of Computer Applications. 3. 10.5120/758-993.
42. UjwalaRavale, NileshMarathe, Puja Padiya, Feature Selection Based Hybrid Anomaly Intrusion Detection System Using K Means and RBF Kernel function, Procedia Computer Science, Volume 45, 2015, Pages 428-435, ISSN 1877-0509,https://doi.org/10.1016/j.procs.2015.03.174.
43. Liu, C., Yang, J. & Wu, J. Web intrusion detection system combined with feature analysis and SVM optimization. J Wireless Com Network 2020, 33 (2020). https://doi.org/10.1186/s13638-019-1591-1
44. Kausar, N., Belhaouari Samir, B., Abdullah, A., Ahmad, I., Hussain, M. (2011). A Review of Classification Approaches Using Support Vector Machine in Intrusion Detection. In: AbdManaf, A., Sahibuddin, S., Ahmad, R., MohdDaud, S., El-Qawasmeh, E. (eds) Informatics Engineering and Information Science. ICIEIS 2011. Communications in Computer and Information Science, vol 253. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-25462-8_3
45. Brao, Bobba&Swathi, Kailasam. (2017). Fast kNN Classifiers for Network Intrusion Detection System. Indian Journal of Science and Technology. 10. 1-10. 10.17485/ijst/2017/v10i14/93690.
46. Wazirali, R. An Improved Intrusion Detection System Based on KNN Hyperparameter Tuning and Cross-Validation. Arab J SciEng 45, 10859–10873 (2020). https://doi.org/10.1007/s13369-020-04907-7
47. Radoglou Grammatikis, Panagiotis&Sarigiannidis, Panagiotis. (2018). An Anomaly-Based Intrusion Detection System for the Smart Grid Based on CART Decision Tree. 10.1109/GIIS.2018.8635743.
48. A. F. A. Pinem and E. B. Setiawan, "Implementation of classification and regression Tree (CART) and fuzzy logic algorithm for intrusion detection system," 2015 3rd International Conference on Information and Communication Technology (ICoICT), 2015, pp. 266-271, doi: 10.1109/ICoICT.2015.7231434.
49. B. S. Sharmila and R. Nagapadma, "Intrusion Detection System using Naive Bayes algorithm," 2019 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), 2019, pp. 1-4, doi: 10.1109/WIECON-ECE48653.2019.9019921.
50. Panda, Mrutyunjaya&Patra, Manas. (2007). Network intrusion detection using naive bayes. 7.
51. Z. Shi and J. An, "An Intrusion Detection System Based on Hadoop," 2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom), 2015, pp. 826-830, doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.162.
52. Vuong, Tuan &Loukas, George &Gan, Diane &Bezemskij, Anatolij. (2015). Decision Tree-based Detection of Denial of Service and Command Injection attacks on Robotic Vehicles. 10.1109/WIFS.2015.7368559.
53. Vuong, Tuan &Loukas, George &Gan, Diane &Bezemskij, Anatolij. (2015). Decision Tree-based Detection of Denial of Service and Command Injection attacks on Robotic Vehicles. 10.1109/WIFS.2015.7368559.
54. Labonne, Maxime. (2020). Anomaly-based network intrusion detection using machine learning.
55. Jair Cervantes, Farid Garcia-Lamont, Lisbeth Rodríguez-Mazahua, Asdrubal Lopez, A comprehensive survey on support vector machine classification: Applications, challenges and trends, Neurocomputing, Volume 408, 2020, Pages 189-215, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2019.10.118.
56. Zeeshan Ali Khan, Peter Herrmann, "Recent Advancements in Intrusion Detection Systems for the Internet of Things", Security and Communication Networks, vol. 2019, Article ID 4301409, 19 pages, 2019. https://doi.org/10.1155/2019/4301409