Journal of Network and Systems Management (2023) 31:50
https://doi.org/10.1007/s10922-023-09742-3

RESEARCH

Mitigating DoS Attack in MANETs Considering Node Reputation with AI

Subrata Joardar[1] · Nilanjan Sinhababu[2] · Soumyodeep Dey[1] · Prasenjit Choudhury[1]

Received: 21 September 2022 / Accepted: 22 May 2023 / Published online: 7 June 2023

Abstract
Mobile Adhoc Network (MANET) is a decentralized and dynamically adoptable network. It is infrastructure-less and hence can be used where a fixed configuration is not possible or required. MANETs have various real-life applications and hence have gained the attention of the research community. Security is an integral part of any computer network system and MANETs are no different. This paper focuses on solving DoS attacks in MANET and shows that a general classification model might fail to identify this kind of attack as these models fail to differentiate between network errors and a real DoS attack. A reputation-based node classification scheme is proposed to improve the identification of real DoS attacks versus any other cause that might not be an attack. Results showed that our proposed reputation-based approach when integrated with any classifier increases its accuracy by around 3.25%. Further, the combined model can block real DoS attacks and allow any other cause which is not an attack.

Keywords Mobile Adhoc Network (MANET) · Node reputation · Machine learning · DoS attacks · Network security

✉ Subrata Joardar
subratajoardar@gmail.com

Nilanjan Sinhababu
nilanjan.thecseboy@gmail.com

Soumyodeep Dey
soumyodeepdey96@gmail.com

Prasenjit Choudhury
prasenjit0007@yahoo.co.in

1 National Institute of Technology, Durgapur, India

2 Indian Institute of Technology, Kharagpur, India

1 Introduction

MANET is an infrastructure-less network on a purely temporary basis, connected by a set of mobile nodes without any centralized system. Applications of MANET have been seen in many fields. Mobile Ad hoc networking helps the military to maintain information networks between military personnel, vehicles, and military information headquarters. Ad hoc networks can be applied in emergency or rescue operations for disaster relief efforts for example in fire, flood, or earthquake and so on. Other commercial applications include for instance ship to ship Ad Hoc Mobile communication and so on. Ad hoc networks can autonomously link immediate and temporary multimedia networks by using notebook or palmtop computers to distribute and allocate information among conference or classroom participants. Besides, it can also be applied to home networks where devices can be linked. Another instance would be a sports arena, watercraft, or tiny aircraft. Short-range MANET can simplify intercommunication between a lot of mobile devices such as a PDA, a laptop, and a cellular phone, and there are a lot of new devices in this for MANETs. MANET, though very popular over a decade, is not available as a standard benchmark and has not received any application in either the business standard or the commercial field. The nature of the mobile environment makes it very vulnerable to an adversary's malicious attacks [1]. The use of wireless links in the network is susceptible to attacks ranging from passive eavesdropping to active interfering. In wired networks, an adversary may gain physical access to the network wires whereas, wireless networks can come from all directions and can target any node. All of these indicate that a wireless ad-hoc network lacks a clear line of defense and that each node must be ready for direct or indirect combat with an attacker. Second, there are various reasons for packet losses in MANET: node-related, congestion-related, and mobility-related [2]. Node-related losses: A node in a forwarding path may refuse to forward routing or data traffic on purpose, either to conserve its limited resources (selfish behavior) or to cause network operation and performance to be disrupted (malicious behavior). Congestion-related losses: Packet losses in this category happen at the MAC layer for a variety of causes. *Queuing problem*: A forwarding node may drop an incoming packet due to high data rates and insufficient link bandwidth, which causes congestion and queue overflow. *Busy channel*: The forwarding node's data channel may be so busy that the number of back offs exceeds the limit, and the packet is discarded. *Link interference*: A data packet may be rejected or discarded due to transmission errors caused by link-related phenomena such as high bit error rate, hidden nodes, and interference. Mobility-related losses: In this category, packet losses can occur at both the MAC and routing layers. *MAC layer*: Packet loss occurs when a packet's next hop is out of range. Because routing information becomes obsolete faster as node mobility increases, this phenomenon is more common in highly mobile networks than in low-mobility networks. *Routing layer*: When a packet reaches the network layer, the routing protocol looks for a valid route and forwards it if one exists. The packet is buffered if there isn't a route to the destination available. A packet is dropped in one of two situations: when it remains in the buffer over the timeout limit, or when the buffer overflow prevents the packet from being buffered.

Third, decision-making in the mobile computing environment is sometimes decentralized and some wireless network algorithms rely on the cooperative participation of all nodes and the infrastructure. Due to the lack of centralized control, adversaries can take advantage of this weakness to launch new kinds of attacks aimed at destroying cooperative algorithms. There are several Intrusion Identification and prevention measures [3, 4], such as encryption and authentication, that can be used in MANETs to reduce intrusions but cannot eliminate them. There are well-organized Intrusion Detection Systems developed for wired networks. But there are no well-designed intrusion schemes for Ad hoc networks. The main difference between wired and Ad hoc networks are infrastructure. While most of today's wired Intrusion Identification schemes depend on real-time traffic analysis, they capture this information by relying on switches, routers, or gateways. This type of equipment is absent in ad hoc networks and causes the most difficult to design good identification schemes. Because of the selfish nature of mobile nodes, it is very difficult to build any scheme. This paper aims to use machine learning techniques to provide an efficient classification of the node which is malicious and the node which is normal in MANET .

1.1 DoS Attack

The wireless nodes are quite prone to be compromised and are particularly weak to different denial of service (DoS) attacks performed by malevolent nodes or intruders. A DoS attack is called a distributed denial of service (DDoS) attack if it originates from multiple distributed sources. A DoS attack is regarded as an attempt to prevent the legitimate use of a service. The main goal of the attack was to temporarily or permanently deny authorized users access to the services and resources. It is commonly carried out by overloading the victim machine or resource with an enormous number of requests making the systems inaccessible [5]. DoS attacks have thus become a major security concern and have attracted the interest of many researchers. However, none of the remedies proposed so far have successfully curbed the impact of the DoS attack in MANET in practical scenarios.

1.1.1 Motivation

If we look at our problem statement from an overall perspective, it can be termed a binary classification problem. The basic approach to solving this would be to classify each node in the network as malicious or normal. But this would be a very generic approach where we would try to apply any machine learning algorithm to solve this binary classification problem. Instead, we try to consider the history of a node before classifying it as malicious or normal. This can also be termed the reputation of a node. This reputation-based approach helps us maintain the reputation of all the nodes in the network and aids us in evaluating their trustworthiness. It helps us counter the various anomalies resulting the selfish and malicious nodes in the network. The intuition behind this idea is to provide an incentive or credit-based mechanism which helps the nodes to cooperate while also improving the overall network performance and functionality by preventing DoS and DDoS attacks. This can be compared to a real-life example of giving loans to people based on their credit scores. A person is given a loan only if his/her credit score is above a certain value. Similarly, each node is assigned a value corresponding to its reputation, and the higher the value, the more would be the node's credibility. Another point of similarity is that irrespective of whether that person has a high or low credit score, he always has an opportunity to increase his credit score and thus become eligible for higher loans. In our case, a node having a low-value reputation can increase its reputation by choosing to participate in ethical and non-malicious activities. This approach helps us in minimizing the cases where a malicious node has been classified as normal, thus protecting from the DoS and DDoS attacks which ultimately would increase the security of the network.

Traditional approaches that are used for DDoS attack detection like Firewalls, filtering techniques, and traceback have many inherent limitations. Machine learning approaches have been used in the recent past to overcome the limitations of traditional approaches while DoS and DDoS attack detection. A node misclassification may happen mainly due to two reasons. The possibility of an ML model being a weak learner and thus giving a greater number of false positives is one of the reasons. Network difficulties impede the different nodes in a MANET design from interacting successfully with one another. This is also one of the possible causes for a node being misclassified. The concern with classifying a normal node as malicious is that we are effectively excluding a possible good node from the network and thus making the MANET network more resource-constrained. On the other hand, when a malicious node gets classified as normal, it can have serious implications on the network and can turn out to be counterproductive for the MANET architecture.

1.1.2 Limitations of Traditional Approaches

MANETs are high in demand and application in today's world owing to their many advantages. One of the main reasons is that its deployment does not require any centralized administration. Even after all that, there are a few challenges like open network architecture, strict constraints for resources, and its highly dynamic network topology which make it vulnerable to external attacks like DoS and DDoS. A system offering security for a MANET architecture should ensure that the services offered to a mobile user are confidential, maintain integrity, and are authenticated. One of the common defense attacks against DoS attacks is a firewall. It is a system for network security that keeps track of and manages all incoming and outgoing network traffic under pre-established security standards.

Firewalls cannot distinguish between normal traffic and DoS attack traffic. Simple rules are followed like allowing some ports or IP addresses which can be counter-productive in case of a resource-constrained environment. Other disadvantages are that it is client-dependent. Firewalls do not prove to be much effective in case they are not up to date. Moreover, many small devices are not computationally adept at employing firewalls.

Filtering is another primary concept that is used to mitigate DoS and DDoS attacks in MANETs. It could be local, global, or statistical. A filter in the local router is installed in case of *local filtering* to stop the malicious nodes. But if the victim's local network can be jammed with enough traffic, the local router can be compromised thus overloading the local filtering. In the case of *global filtering*, the idea is to prevent any accumulation of malicious packets in each time frame. Filters are installed all over the Internet and when any victim detects an attack, it shares this information with all the other nodes. This can result in the malicious nodes being stopped early. However, this attack cannot be considered reliable since sometimes the router can get compromised by the continuous flow of packets, thus causing a DoS attack. Another filtering approach is *Statistical filtering*. Here in this approach, the statistics of a packet are observed closely to classify its behavior as normal or malicious. The packets that are classified as malicious are dropped by the filter. This again can be a problem in a resource-constrained environment. Another major limitation of it is that it is a cluster-based routing protocol filtering mechanism.

Traceback is another approach to detect DoS attacks. Here the main aim is to trace the intruder back to the zombie computers and thus help in identifying the source of the attacker. Cost management, low accuracy of results, and slow tracking speeds are some of the drawbacks of these traceback schemes. This becomes ineffective since the attacker moves to another position, owing to the high mobility of nodes in the MANET before the attacker is traced.

Pushback is another approach where routers are enabled to identify the high bandwidth aggregates that contribute to the high congestion rate and help limit it. But the pushback approach is unable to work in non-contiguous deployment and thus unable to stop the DoS attacks that do not overcrowd the core routers.

1.1.3 Why Limitations Of Traditional Approaches can be Mitigated Using Machine Learning Approaches

Currently, a lot of research has been done to mitigate DoS and DDoS attacks using various traditional machine learning algorithms. Since detecting malicious nodes in a manet architecture is predominantly an anomaly detection-based problem, machine learning algorithms perform well in such scenarios.

Xiao et al. [6] presented a detection approach that exploits correlation information of the training data to improve classification accuracy and reduce the overhead caused by the density of training data. The approach is based on CKNN (k-nearest neighbors traffic classification with correlation analysis) and performs efficiently to detect DDoS attacks. Agarwal et al. [7] proposed a machine learning approach using support vector machine (SVM) to predict the number of zombies in a DDoS attack. Saad et al. [8] applied and compared the performance of five different machine learning algorithms—support vector machine (SVM), artificial neural network (ANN), nearest neighbors classifier (NNC), the gaussian-based classifier (GBC), and naïve Bayes classifier (NBC) to detect p2p bots which are used to generate spam and carry out DDoS attacks. Here the command-and-control phase for detecting DDoS attacks before they are launched was studied. Sambang et al. [9] study the problem of DDoS attack detection in cloud environments and build a machine learning model using multiple regression analysis to predict DDoS and bot attacks by choosing the most important features in the cicids 2017 research dataset. Fadlil et al. [10] proposed a DDoS attack detection method based on network traffic activity that was statistically analyzed using the gaussian naïve Bayes method. This approach predicted the existence of DDoS attacks based on the average and standard deviation of the network packets according to the gaussian method. Yi-Chi Wu et al. [11] proposed another DDoS detection system that uses a decision tree algorithm on 15 different attributes to detect abnormal traffic flow. It also traces back the attacker's locations with a traffic-flow pattern matching technique. Suresh et al. [12] study one of the major limitations in statistics-based detections is that they can only be simulated as a uniform distribution and it is not possible to find out the normal network packet distribution.

In traditional approaches, the whole dataset is used to make a prediction, whether there is a DDoS attack or not. In the case of machine learning approaches, the whole dataset is divided into two parts, the training data which is used to train the model, and the testing data which is used to observe the performance of the model on unseen data. This ensures that the model is less biased. Machine learning models offer a new glimmer of hope as they can address the gaps in traditional DoS and DDoS detection algorithms, by performing well on even new and unseen DDoS attacks.

1.1.4 Limitations of the Machine Learning Approach

Although the advantages of Machine learning approaches are discussed in the previous section, it becomes particularly difficult to extract and select a valid number of independent features for building an efficient machine learning model to identify DDoS attacks. Many variables can be used to characterize network traffic patterns, and if the task of feature reduction and extraction is not done properly, it may affect the time required in to train and test the model. Thus, the task of feature engineering holds special importance in this domain as it can help in differentiating between the normal and the malicious nodes. As already discussed previously, Saad et al. [8] used a machine-learning approach to detect P2P botnets before they are even launched. The major limitation that came along with this work is that it can only detect a single compromised host and is unable to detect a whole BotNet. We have already seen that machine learning algorithms perform exceptionally well while detecting DoS and DDoS attacks and produce high degrees of accuracy. But they have their own set of limitations. Often, most machine learning algorithms require a lengthy training period and even if they give good results, they cannot be used in real-time. Moreover, these algorithms are highly demanding in terms of computational expenses.

Due to all these limitations, there is a high chance of misclassifications. And, as per the general implementation of these algorithms misclassification will result in eliminating either a good node or accepting an attack. Further, when considering real-time implementation, where the number of processes keeps on increasing, makes these techniques are infeasible for machine learning algorithms to be used.

1.1.5 Scope

We identified the following scope of research for mitigating DoS Attack.

1 As per the literature survey, every work done in this domain is based on the generalized nature of the security, i.e., they assume that every system has similar security needs. But the security of a system has a very personalized aspect with varying requirements. Considering a methodology for a highly secured system is missing in the literature.
2 A methodology that can be adopted on any system is still not available.
3 Most of the work in the literature has directly classified the node, which poses a severe problem in case of misclassification.
4 Existing work does not consider different costs associated with the misclassifications. They assign equal weightage for all the misclassification errors. A non-malicious node that has been correctly classified and a malicious node that has been incorrectly classified are given equal importance.

1.1.6 Objective

The objective of the paper is outlined below:

1. Design a reputation-based scheme for the MANET environment.
2. To find variation in the model performances for general algorithms i.e., SVM and DNN versus the integration of reputation schemes with those algorithms i.e., RSVM and RDNN.
3. To minimize the most significant error based on different costs associated with the misclassification errors.

2 Literature Review

2.1 Old Methodologies to Obstruct DoS Attacks in MANET

Mobile ad-hoc networks (MANETs) are particularly vulnerable to denial of service (DoS) attacks originating through compromised nodes or intruders. Goals of a DoS attack is to degrade or deny normal facilities for legitimate nodes through the distribution of huge traffic to victims which affects the network, host, and resources in different ways. There are many approaches to handle this attack such as traditional methods and other specific methods. Here we survey papers that presented various techniques on how to obstruct DoS attacks in MANET Table 1.

2.2 Machine Learning Approach

The purpose of the review is to understand the current trends for DDoS anomaly detection in MANETs Table 2. It is clear from this review that traditional techniques like firewalls and filtering are good until a new anomaly appears. Since these

Table 1 Survey in considering to obstruct DoS attacks in MANET in old methodologies

Title name	Description	Approach/mechanism	Drawback
Fully distributed dynamicallyconfigurable firewall to resist DOS attacks in MANET [13]	Presented a distributed dynamically configurable firewall architecture that uses ingress and egress filtering to resist the DoS attacks	Firewalls	Firewalls cannot distinguish between normal traffic and DoS attack traffic. Next, due to the mobility of the node firewall cannot be sufficient for MANETs
A cooperative approach for understanding behavior of intrusion detection system in mobile ad hoc networks [14]	IDS collects, monitors, and analyses audit data to find any anomalous attempts	An intrusion detection system (IDS)	Many false alarms, fidelity problems, and overhead problems
Framework for statistical filtering against DDoS attacks in MANETs [15]	Statistical filtering is proposed by using traffic profiling for filtering and DDoS attack detection	Filtering	Not reliable as sometimes the packets can overwhelm the router and cause a DoS attack
A review on different Intrusion Detection Systems for MANET and its Vulnerabilities [16]	Watchdog tagged the node as a misbehaving node if it fails to forward the packet to the next node	Watchdog/Pathrater	Watchdog cannot detect malicious nodes in the presence of receiver collisions, limited power for transmission, and false misbehavior reports
Hotspot-based traceback for mobile Ad Hoc networks [17]	This method helps to identify the source of an attack i.e. physical location of the attacker	Traceback	Since nodes are moving so the attacker could move to another position before the tracing process is finished
DoS pushback. ENCYCLOPEDIA OF CRYPTOGRAPHY AND SECURITY [18]	In this mechanism routers are enabled to identify the high bandwidth aggregates that participate in the congestion rate and help to limit them	Pushback	Pushback is unable to work in non-contiguous deployment and unable to compromise attacks that do not overcrowd the core routers
Detection and prevention of denial of service (DoS) attacks in mobile Ad Hoc networks using reputation-based incentive scheme [19]	This method helps to prevent DDoS attacks by providing cooperation among nodes based on incentive mechanisms	Reputation-based incentive mechanism	The cluster heads were assumed to be stationary and only the nodes of the cluster could move freely. In addition, scalability is not considered in this approach

Table 1 (continued)

Title name	Description	Approach/mechanism	Drawback
Security through collaboration and trust in MANETS [20]	The main objectives of this framework are to support localized control and relationships by binding public keys to allow the access control process without complex security authentication procedures	Trust management	A trust calculation done on a certain node by another node raises the resource cost. Unfortunately, these resources are limited in MANETs
Mitigating denial-of-service attacks in MANET by distributed packet filtering: A game-theoretic approach [21]	This method is based onusing digital signatures to apply verification of legitimate packets. There is a penalty for the forwarders of bad packets and also a reward system for the forwarders that verify packets that are gained as a credit	Game theoretic approach	The signature-based defense is prone to replay attacks

Table 2 Survey in considering DDoS attack detection using a machine learning approach

Title name	Description	Approach/mechanism	Drawback
The DDoS attacks detection through machine learning and statistical methods in SDN [22]	To detect DDoS attacks in an SDN environment that depends on three parts collector, entropy-based, and classification sections based on machine learning algorithms (Random Tree, REPTree)	Dataset used ISCX-SlowDDoS-2016 and the machine learning algorithm used Random Tree	Setting a threshold value that involves several statistics is challenging
Detection of known and unknown DDoS attacks using artificial neural networks [23]	Based on a study of each TCP/UDP/ICMP protocol's features through training of an ANN algorithm to identify DDoS attacks	Dataset used UNB-ISCX and the machine learning algorithm used Neural Network	the method needs to distinguish packet protocol, which is complex and inefficient
Detection of distributed denial of service using deep learning neural network [24]	It reduces the classification error with minimal cost and it improves the accuracy of detection	Dataset used KDD Cup and the machine learning algorithm used deep learning neural network	This model cannot adopt SVM and KMC for real-time applications
Mining based detection of botnet traffic in network [25]	Compare several different machine learning algorithms in the context of network traffic classification	Dataset used CTU-13 and the machine learning algorithm used SVM, naive Bayes, decision trees, and neural networks	Being aimed at a comparison rather than the optimization of a specific approach lacks an accurate feature selection
Anomaly-based intrusion detection through k-means clustering and Naive Bayes classification [26]	Semi-supervised learning (halfway between classification and clustering) is used here. The additional information from the labeled data is combined with the unlabeled data	Dataset used UNB-ISCX and the machine learning algorithm used k-means clustering + Naïve Bayes	High false-alarm rates and lack of accuracy in the detection procedure
A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning [27]	The IDS implemented in the proposed architecture was trained with a set of machine learning algorithms	Dataset used CIC DoS-2017 and the machine learning algorithm used RT, REP Tree, Random Forest, and SVM	The collection of traffic features in small-time intervals increase processing and communication overhead

techniques rely on the entire signature of the attacks, therefore any deviation in the signature will allow the attackers to gain access to the system. This problem is solved in the literature using machine learning approaches, where the algorithms learn the signature and try to predict the possibility of a new signature being genuine or an anomaly. We found that the machine learning approaches learn the difference in the signature of a genuine or an attacker node but never consider a node's history in the prediction. This leads to a hard classification of a node and allows both Type-I and Type-II errors to occur. As per our observation, an improvement to the current work can be made by performing a soft classification considering the history of a node. Further, the degree of severity of Type-II errors in this field is higher than that of Type-I errors, therefore we may introduce a bias to the model to make fewer Type-II errors.

3 Proposed Methodology

When initially a node is created, it is assigned a neutral reputation (0.5RP), where the reputation value ranges between 0 and 1. Since every system is different in terms of security requirements, we provide a filter hyperparameter (ranging between 0.25RP and 0.75RP) that can control the incoming packet from a node. For example, a system that does not care much about security can lower this hyperparameter and allow packets with low reputation nodes. In contrast, a sensitive system with a requirement of high security should take a higher value of this hyperparameter to wait for the node to achieve a higher reputation. From the perspective of the nodes, their reputation is decided by the host that they are interacting with, making this scheme dynamic. e.g., consider a situation where the system (A) threshold is 0.55RP and a new node (B) arrives in the network with a reputation of 0.50RP. Now, B tries to send some packets to A, but due to a higher threshold, instead of accepting or rejecting the packet directly, A holds the packet in a secure buffer and pings B for an updated reputation score. From here two things can happen. First, if the reputation score of B's is not able to cross the threshold, then after the timeout the packet is discarded. Secondly, if B in the meantime sends some packets to system C with a 0.45RP requirement, the packets after getting accepted will increase the reputation of B. After these increments, if the reputation of B is equal to or more than 0.55RP then the packet is accepted by A and forwarded to the next firewalls.

3.1 Justify the Rationale Behind our Approach

Our approach is based on a reputation-based framework where nodes maintain the reputation of other nodes and use it to evaluate their trustworthiness. This is done for detecting anomalies arising from a few malicious and selfish nodes in a MANET architecture. The reputation value of a node shows how reliable it is, based on its history, and aids in the process of decision-making. The anomalies can arise from mainly two kinds of activities—selfish behavior, for example, nodes wanting to save power. The other activity is malicious behavior where the node is primarily concerned with attacking and damaging the network, as in the case of DoS and DDoS attacks. To counter these misbehaviors, an incentive should be provided to all the nodes, so that they can cooperate. This mechanism ensures that even the selfish nodes, which behave in a way to maximize their benefits, also make the most out of the cooperation among the various nodes in the MANET architecture. The main intent of this reputation-based approach is to enable the nodes to distinguish between the trustworthy and the untrustworthy nodes. The approach encourages the nodes to refrain from malicious activities and thus collaborate with the other nodes in the architecture to build on its reputation value. The type of supervised machine learning algorithm used to classify between a normal and a malicious node is irrelevant since this approach is independent of the classifier used. There can be various ways of initialization of the reputation values of the nodes. It can be initialized to zero, meaning that all the nodes are considered untrustworthy in the beginning. They can also be assigned a maximum value of reputation at the start, meaning that all the nodes are considered trustworthy. In our approach, we have tried to take the middle ground by assigning a neutral value of reputation to each node, which signifies that the nodes are neither considered trustworthy nor untrustworthy in the beginning.

3.2 Advantages of the Proposed Reputation-Based Scheme

The packet is temporarily held in a secure buffer because this method does not accept or reject it directly. The message is buffered when an attacker node sends a packet to the system. The packet is discarded after the timeout because the attacker node's reputation score is gradually decreasing and the attacker node cannot let its reputation value dip below the threshold value.

This scheme doesn't instantly reject a packet from a non-attacker node due to misclassification. Since the non-attacker node's reputation score increases over time, the value may surpass the threshold, allowing the packet to be accepted later. In this scheme, if a packet is rejected or accepted due to network traffic errors, it will be remedied later since the reputation value of nodes will not be harmed.

4 Experiment

In the following sections, the experimental setup has been mentioned followed by the description of the Kitsune Network Attack Dataset. The steps undertaken to clean the dataset have also been mentioned where we talk about the various preprocessing steps which include filtering, missing value handling, and how different anomalies have been dealt with. The dataset obtained is then split into the training and the testing data using the train test split function of the *scikit-learn* library. Further, all the classification algorithms (namely Support Vector Machine, Deep Neural Network, Reduced Support Vector Machine, and the Reduced Deep Neural Network) that have been applied to the dataset Fig. 1, have been discussed in detail. Finally, the results obtained from each of these machine learning models have been analyzed which led us to a few conclusions.

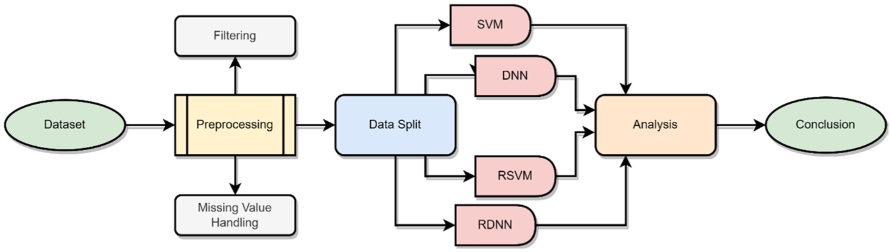


Fig. 1 Experimental flow diagram

4.1 Experimental Environment

4.1.1 The Hardware Specification of the System used is as Follows

OS-Windows 10 Professional, CPU-AMD® Ryzen™ 7-3700X Processor, RAM-32 GB DDR4, GPU-NVIDIA GeForce® GTX 1080 Ti.

4.1.2 The Software and APIs of the Experiment are Given Below

The Windows version of the Python-64 Bit with IPython notebook [28] is used for building the models. Important APIs used in the experiment include TensorFlow [29], packages from NumPy [30], scikit-learn [31], and Matplotlib [32]. NVIDIA CUDA Version 9.1 for Windows environment is used for enabling GPU computing.

4.2 Dataset Description

We have used Kitsune Network Attack Dataset [33] for our experiment. Kitsune is an online, unsupervised, and efficient ANN-based network intrusion detection system (NIDS). Kitsune is made up of a collection of tiny neural networks (autoencoders) that have been trained to replicate (reconstruct) network traffic patterns and whose performance increases over time. These are cybersecurity datasets containing nine distinct network attacks on a commercial IP-based surveillance system and an IoT network. The dataset contains attacks including botnets, MitM, DoS, and reconnaissance. These datasets are downloaded from UCI. The number of instances of these datasets is 27170754 and the number of attributes is 115. For our experiment, we have used 2771275 instances and 115 attributes.

The scatter plot in Fig. 2 represents the attacker and the non-attacker data at the kth dimension. From the visual inspection, it is visible that there is a possibility of classification using a hyperplane. Hence, machine learning classifiers like SVM can be used to solve this problem. Further, deep learning classifiers can also be implemented to check for improvements.

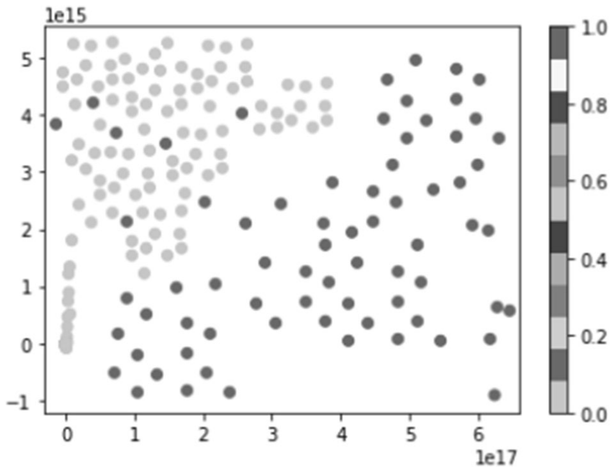


Fig. 2 Dataset scatter plot

4.3 Preprocessing

In the preprocessing phase, we analyzed the entire dataset for

1. *Any missing values*: Values not captured in a dataset are known as missing data. They can range from a single value missing from a single cell to an entire observation being lost (row).
2. *Header anomaly*: Data preprocessing requirements are reduced by anomaly detection based solely on header information. Because headers make up a tiny portion of overall network data, they take fewer resources (CPU, memory, and storage) to process than entire packet payloads.
3. *Specification anomaly*: Anomaly detectors based on specifications take advantage of the fact that protocols change far more slowly than attackers. As a result, modeling protocols rather than constantly establishing signatures for the latest malicious code should be easier.

After anomaly detection is done and the dataset is optimized for any anomaly or sparsity, we use sampling techniques to sample the labeled data.

4.4 Classification

4.4.1 Support Vector Machine (SVM)

SVM is a supervised machine learning algorithm that is used for classification, regression, and outliers detection. SVM works well in high-dimensional spaces; it is still effective when the number of dimensions exceeds the number of samples. It is memory efficient because it uses a subset of training points (called support vectors) in the decision function. The decision function can use a variety of Kernel functions. Common kernels are included, however custom kernels can also be specified according to the data.

Various papers have been published where the detection of malicious attacks in MANET using machine learning approaches being used. Some of them used SVM-based methods. SVM has been used to detect black hole attacks in MANETs using the AODV protocol [34]. Three performance indicators, namely PDR, packet modification rate, and packet misroute rate, are used in the proposed SVM-based technique to classify the type of nodes. The numbers of sent, modified, and misrouted packets are used to produce these metrics. The SVM-based strategy outperformed the previous method, according to the findings. However, the proposed SVM's explanation is ambiguous. Aside from that, the simulation outcomes are ambiguous. The SVM-based algorithm discovers more harmful nodes than the previous method, but no explanation is provided. A new architecture for intrusion detection in MANETs has been suggested that maximizes detection accuracy by employing a machine learning technique [35]. They proposed a feature selection technique namely a rough set and SVMs were utilized in this study for data reduction and classification, respectively. To lower the complexity of SVM, the rough set reduces the size of features Fig. 3.

Although SVM works well in many domains, it is not suited for extremely large data sets. When there is more noise in the data set, such as when target classes overlap, SVM does not perform well. The SVM will underperform when the number of features for each data point exceeds the number of training data samples.

SVM classifies nodes as either attacker or normal. Thus, class label $y_i \in$ {attack, normal}. Given the training datasets $(x_i, y_i), 1 \leq i \leq n$, x_i is used for the training. The objective is to find the hyperplane that offers a maximum margin between the two classes.

The equation of hyperplane is given as follows:

$$g(x) = w^T X + b \tag{1}$$

where X is the input feature vector, w is the weight vector which represents the orientation of the hyperplane in space. And, b is the bias vector which represents the position of the hyperplane in space.

The equation $g(x)$ given above divides the space into two subspaces. For a binary classification problem, where there are two classes (let us assume them to be class $C1$ and class $C2$), one of the subspaces denotes the space for $C1$ and the other subspace is for class $C2$. Mathematically for a point $x_{1,}$ it can be written as:

$$g(x_1) = w^T x_1 + b > 0 \tag{2}$$

such that: $x_1 \in C1$

$$g(x_1) = w^T x_1 + b < 0 \tag{3}$$

such that: $x_1 \in C2$

Let d be the measure of distance X to the separating plane. So, we can say that

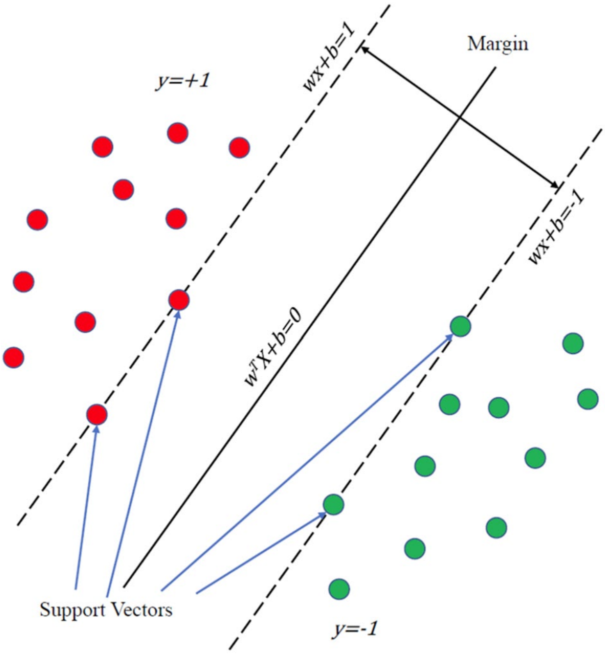


Fig. 3 Support Vector Machine (SVM)

$$w^T x + b \geq d \tag{4}$$

or,

$$\frac{w.x + b}{||w||} \geq d \tag{5}$$

where, $||w||$ is the norm of w. such that,

$$w.x + b \geq d * ||w|| \tag{6}$$

We know that the value of d * $||w||$ is 1. Therefore, the equation can be rewritten as:

$$w.x + b \geq 1 \tag{7}$$

if $x \in C1$

$$w.x + b \leq -1 \tag{8}$$

if $x \in C2$

To reduce the expression down to one term, we introduce another term y_i which represents the class of the *ith* point.

So,

$$y_i\left(w.x_i + b\right) \geq 1 \tag{9}$$

We can rewrite the above equation as

$$y_i\left(w.x_i + b\right) = 1 \tag{10}$$

This equation is only valid for support vectors. Support Vectors are the data points or vectors that are closest to the hyperplane which affect the position of the hyperplane. The margin *d* needs to be maximized in Eq. 5. This is because our main objective is to find the hyperplane that offers maximum margin between the two classes. Maximizing the margin prevents over-fitting in high dimension input spaces, which ultimately leads to good generalization capabilities. This can be achieved by the maximization of the value of bias vector *(b)* or the minimization of the norm of *w (||w||)*.

The equation which needs to be minimized is given as:

$$\Phi(w) = \frac{1}{2}||w||^2 \tag{11}$$

The above optimization needs to be achieved under a given constraint, given by the Eq. 10:

$$y_i\left(w.x_i + b\right) = 1 \tag{12}$$

Since this is a constraint optimization problem, it can be converted into an unconstrained optimization problem by using the Lagrangian Multiplier

$$L(w, b) = \frac{1}{2}||w||^2 - \sum \alpha_I \left[y_i\left(w.x_i + b\right) - 1\right] \tag{13}$$

α_i in Eq. 13 denotes the Lagrangian Multiplier.

$$L(w, b) = \frac{1}{2}||w||^2 - \sum \alpha_i y_i\left(w.x_i\right) - \sum \alpha_i y_i b + \sum \alpha_i \tag{14}$$

To minimize the above expression, we find the gradient of *L*, by taking the partial derivatives of *L* w.r.t. the variables *b* and *w* and equating them to be zero.

$$\frac{\partial L}{\partial b} = -\sum \alpha_i y_i = 0 \tag{15}$$

$$\sum\nolimits_{i=1}^{m} \alpha_i y_i = 0 \tag{16}$$

Here *m* is the number of training samples.

$$\frac{\partial L}{\partial w} = w - \sum \alpha_i y_i . x_i = 0 \tag{17}$$

$$w = \sum_{i=1}^{m} \alpha_i y_i x_i \tag{18}$$

Substituting Eq. 16 and Eq. 18 in Eq. 13:

$$L(w, b) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y_i . y_j (x_j x_i) \tag{19}$$

The above Lagrangian expression needs to be maximized with different values of α.

Lagrangian multipliers are always non-negative.

So, $\underline{\alpha}_i \geq 0$, which satisfies the condition

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{20}$$

For an unknown feature vector Z:

$$D(Z) = \sum_{j=1}^{m} \alpha_j y_j x_j . Z + b \tag{21}$$

Only the sign of the above expression is important for finding out which class it belongs to.

$$L(w, b) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y_i . y_j K\left(x_i x_j\right) \tag{22}$$

Since the dataset used in this experiment is non-linear and cannot be directly used in SVM, we need to use a kernel function. For most of the non-linear data, polynomial and RBF kernels are used. For our experiment, we tested with both polynomial and RBF kernels and found that the RBF kernel results were around 37% more accurate. This means that the polynomial kernel was unable to transform data properly in the hyperplane. So, we selected and used the RBF kernel function for SVM for the rest of the work.

The Radial Basis Function kernel or the RBF kernel is the most powerful form of the kernel since it contains an exponent term. The exponentiation of a value gives a polynomial term of infinite dimensions. It helps in fitting a generalized form of a curve on the most complex datasets. Mathematically it is represented as follows:

$$K\left(x_i, x_j\right) = exp(-\frac{||x_i - x_j||}{2\sigma^2}) \tag{23}$$

$\|\, x_i—x_j \,\|$ represents the Euclidean distance between the two points x_i and x_j.$\boldsymbol{\sigma}$ is the variance which can also be treated as a hyperparameter.

4.4.2 Deep Neural Network (DNN)

Neural Networks have been designed to imitate the working of the human brain. They consist of simple processing units called nodes. A collection of nodes together consists of a layer of a neural network. The number of layers of a neural network denotes its depth. Any neural network with more than two hidden layers (except the input layer and the output layer) is called a Deep Neural Network (DNN). Each layer in a DNN is a function (also called an activation function). Activation functions play a very crucial role in determining the output of a node, given a set of inputs. Some of the most popular activation functions are tanh (hyperbolic tangent function), relu (rectified linear unit), sigma (sigmoid function), etc.

The fundamental difference between any traditional Machine learning algorithm and a Deep Neural Network is that the former works better on smaller datasets. As the amount of data keeps on increasing, the performance of DNNs also keeps getting better.

There are various kinds of neural networks that are available and they have their own set of applications. Some of the widely used neural networks are Radial Basis Function Neural Networks (RBFNN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Generative Adversarial Network (GAN) and many more. CNNs are mostly used for image processing, classification, segmentation, recognition, etc. It mainly comprises a few convolution layers and a few other layers like pooling layer and softmax layer. RNNs are the go-to algorithm for all kinds of sequential data. It is mainly used for sentiment classification, sequence labeling, predicting the next word, and other language modeling tasks. We have used simple Feed Forward Neural Networks in our experiments to predict whether a given node is malicious or not.

4.4.2.1 Literature Review of DNN for a Similar Problem [36] applied four different deep learning approaches for Intrusion Detection System (IDS) in MANET architectures and then compared their results. These four approaches were—Convolutional Neural Networks (CNN), Inception Convolutional Neural Networks, Bidirectional Long Short-Term Memory (Bi-LSTM), and Gated Recurrent Units (GRU). The first two were CNN based Intrusion Detection Systems and the last two were RNN based Intrusion Detection Systems. All these four models were tested on the NSL-KDD dataset and their Precision, Recall, and Accuracy values were compared to determine which model performed better.

[37] uses a hybrid model approach for the exact classification of malicious network flow from the packets. The main idea behind the approach is to use an autoencoder-based deep neural network algorithm to separate malicious nodes from non-malicious ones. The model relies on sampled network flow data. The autoencoder-based approach helps in avoiding overfitting to pre-defined malicious patterns,

[38] uses a hybrid deep neural network approach to detect Low-rate Denial of Service (DoS) attacks in fluctuating legitimate traffic. It uses a one-dimensional Convolutional Neural Network and a Gated Recurrent Unit to detect DDoS attacks in fluctuating HTTP traffic.

[39] incorporates a DDoS detection framework, a Bidirectional Long Short-Term Memory (Bi-LSTM), a Gaussian Mixture Model (GMM), and incremental learning. This framework helps to counter the Open Set Recognition (OSR) problem in DDoS attacks. The Bi-LSTM layer helps in capturing the essential characteristics of the DDoS traffic, especially the time domain correlations whereas the GMM in

the architecture helps to differentiate between the trained samples and the novel instances.

[40] combines a Long Short-Term Memory (LSTM) and Bayes approach and refer to the as LSTM-BA to propose a novel DDoS detection algorithm. The LSTM layer in the model helps identify parts of the DDoS attack which possess high-confidence outputs and for those outputs with low-confidence the Bayes method is used to improve the accuracy.

The above-mentioned papers and all the related work that has been done in building Intrusion Detection Systems using the deep learning approach use the same concept of increasing the model accuracy by making correct predictions which are achieved by building a more complex architecture or by introducing some new deep learning-based model. Our approach differs from these approaches by not just focusing on increasing model accuracy but also considering the history of a node while making a prediction.

Like ML models, DNN models will also suffer from the fact that classifications will be instantaneous without consideration of the history of any node. It is unlike our approach where the reputation of a node is considered to determine whether a given node has malicious intent or not. In this Deep Neural Network approach, we feed data about a node to the input layer and after passing through many hidden layers or abstractions it finally passes through a softmax layer which gives a probabilistic output ranging between 0 and 1. Where the values 0 and 1 represent the class as either genuine or an attacker respectively.

Let us consider a set of inputs $< x_1\ x_2\ x_3\ \ldots\ x_m >$ of size m. A weight is assigned to each connection between an input vector and a single neuron of the hidden layer. For example, the weight assigned to the connection between the first input vector and the first neuron of the first hidden layer would be denoted as w_{11}. Similarly, the weight assigned to the connection between the second input vector and the first neuron of the hidden layer would be denoted as w_{12}, and so forth.

The output of a single neuron in a hidden layer is calculated by the matrix multiplication between the input feature vector and the weights. Let us denote the output by z.

Therefore,

$$z = w_{11}x_1 + w_{21}x_2 + w_{31}x_3 + \cdots + w_{m1}x_m \tag{24}$$

The result of this calculation would give us the output of the first neuron of the first hidden layer. We can also denote it by the matrix multiplication between the weight vector and the input feature vector.

$$z = w^T.x + b \tag{25}$$

Here W represents the feature vector representation of the weight vectors.

$\mathrm{W}: << \mathrm{w}_{11}.\mathrm{w}_{12}, \ldots\ldots \mathrm{w}_{1\mathrm{n}} >, < \mathrm{w}_{21}.\mathrm{w}_{22}, \ldots\ldots \mathrm{w}_{2\mathrm{n}} >, < \mathrm{w}_{31}.\mathrm{w}_{32}, \ldots\ldots \mathrm{w}_{3\mathrm{n}} >, \ldots\ldots\ldots < \mathrm{w}_{m1}.\mathrm{w}_{m2}, \ldots\ldots \mathrm{w}_{\mathrm{mn}} >>$

X represents the input feature vector.

$$X : < X_1 \cdot X_2 \cdot X_3 \ldots X_m >$$

b is nothing but the bias vector.

The neuron calculates the weighted average of the values using the current value of input vector X. The values of the weight vector and the bias vector in each layer keeps getting updated in each iteration and thus the predicted output from the neural network also gets more accurate after every iteration. To keep the output of a neural network relevant we need to introduce non-linearity into the architecture, otherwise, it just becomes like any other linear regression model. Therefore, we need to introduce the concept of the activation function. Its role is to calculate the weighted sum of its inputs and add the bias term. The most frequently used activation functions are:

I) Step Function

$$f(x) = 1 \tag{26}$$

if $x >= 0$

$$f(x) = 0 \tag{27}$$

if $x < 0$

Gives an output of either 0 or 1.

II) Sigmoid Function

$$f(x) = 1/(1 + e^{-x}) \tag{28}$$

Gives an output in the range of 0 to 1.

III)ReLU (Rectified linear Unit)

$$f(x) = max(0, x) \tag{29}$$

IV)Hyperbolic tangent Function

$$f(x) = tanh(x) = \left(\frac{2}{\left(1 + e^{-2x}\right)}\right) - 1 \tag{30}$$

Gives an output in the range -1 to 1.

Many other mathematical functions are used as activation functions in neural networks. The above-mentioned ones are only a few of them. For the sake of clarity let us denote the activation function being used in a hidden layer of a neural network as g. Thus, the output coming out of a neuron in a hidden layer can be given as:

$$a_i = g\left(z_i\right) \tag{31}$$

Since there are many other hidden layers in a neural network, Eq. 25 and Eq. 31 can be generalized for all the layers as follows:

$$z_i^{[l]} = W_i^T a_i^{[l-1]} + b_i^{[l]} \tag{32}$$

$$a_i^{[l]} = g^{[l]}\left(z_i^{[l]}\right) \tag{33}$$

The superscript l here denotes the lth layer of the neural network. $a^{[0]}_i$ can also be written as x_i.

The result generated from the output layer is interpreted using a softmax layer. If it is a binary classification problem, like ours, the softmax layer gives an output of either 0 or 1 meaning attacker or non-attacker.

But even so, the above-mentioned steps only constitute the forward propagation part of the neural network. Let us denote the output generated after an iteration to be ŷ. While training the neural network we already have the original result with us. Let us denote it by y. By determining how different the predicted output is from the original output, we can calculate the loss incurred. Using this loss value, we can update the weight and bias vector parameters of each layer. By doing that we are ensuring that in the next iteration the loss incurred would be comparatively lower. This whole process is termed backpropagation and this is what makes the neural network architecture so effective.

The loss between the predicted output ($\hat{y}$) and the correct output (y) is calculated using the binary cross entropy function. It is given as follows:

$$L(\hat{y}, y) = -(y log\, \hat{y} + (1 - y) log\, \hat{y}) \tag{34}$$

It is not mandatory to use binary cross entropy as our cost function. We can also use Mean Absolute Error (MAE) or Root Mean Square Error (RMSE) as our loss functions.

Suppose there are t training samples. The Cost function is synonymous with the loss function. The only difference is that the Cost Function (let us denote it by J) is the average of the loss errors of all the training samples. It can also contain a regularization term. It is a function dependent on two variables—W and b. Therefore, mathematically it can be expressed as:

$$J(W, b) = \frac{1}{t} * \sum_{i=1}^{t} L\left(\hat{y}^{(i)}, y^{(i)}\right) \tag{35}$$

The next step in backpropagation involves the calculation of the gradient of cost function J concerning its dependent variables W and b. The objective of this step is to update parameters W and b such that the loss function is minimized in each iteration. This is done with the help of the gradient descent method which proceeds by calculating the partial derivative of the cost function of the parameters W and b. By finding the partial derivative of Eq. 35 with respect to W we get the following result: -

$$dW^{[l]} = \frac{\partial J}{\partial W^{[l]}} = \frac{1}{t} * dz^{[l]} a^{[l-1]T} \tag{36}$$

The partial derivative of Eq. 35 with respect to b is:

$$db^{[l]} = \frac{\partial J}{\partial b^{[l]}} = \frac{1}{t} * \sum_{i=1}^{t} dz_i^{[l]} \tag{37}$$

Using Eq. 36 and Eq. 37 we can update parameters W and b as follows:

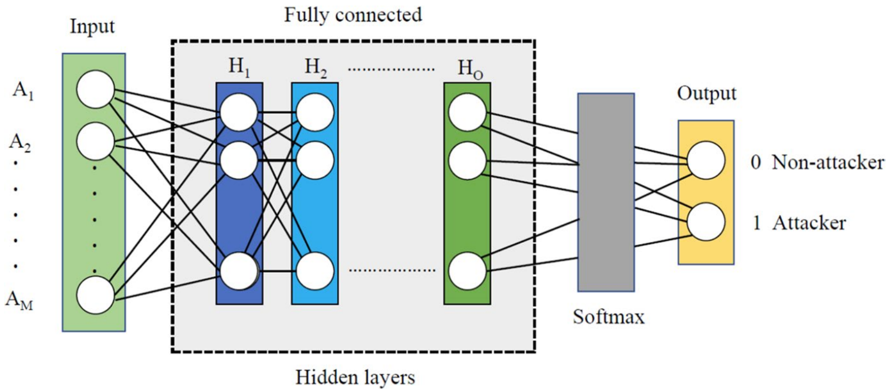


Fig. 4 Deep neural network architecture

$$W^{[l]} = W^{[l]} - \alpha.dW^{[l]} \tag{38}$$

$$b^{[l]} = b^{[l]} - \alpha.db^{[l]} \tag{39}$$

α is just a scalar constant in these equations. Technically, it is termed as the learning rate whose value determines how fast the neural network learns and updates its parameters W and b. It is a hyperparameter and the more optimized its value, the faster will it hit the minimum. It shouldn't be either too high or too low.

The DNN architecture is given in Fig. 4. In this architecture, the input layer consists of 115 neurons for each of the attributes. Then, the total number of hidden layers is selected as two. Both the hidden layer consists of 57 neurons (approximately half of the input neurons) of RELU activation units. Finally, the output neurons consist of two classes, one for classifying attackers and the other for non-attacker.

4.4.3 Reputation-based Classifiers

In our proposed approach, in addition to the existing fields of a node, we introduce two additional fields RP in Fig. 5. The figure represents the outline of the model formation when using a reputation scheme. The reputation strategies are maintained by Reputation Processing System (RPS), while the basic classification is done using the standard node metadata MD in Fig. 5. The first field is the reputation threshold (T_m). The value of this field might be modified by the node itself as per its requirements. The value ranges from 0 to 1. In a place where we need more security, then the reputation threshold may be enhanced. Vice versa in a place where we need less security, then the reputation threshold may be diminished. The second field stores the reputation score of the node. The value of this field might be modified by the behavior of the node in the networks, not by the node itself. Through simulation, we try to find how this reputation score changes and that has been shown in the given three graphs with different 3 reputation thresholds 0.25, 0.50, and 0.75.

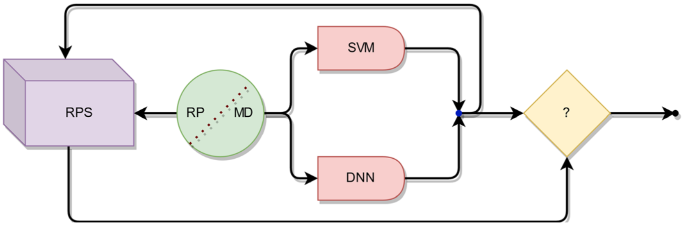


Fig. 5 Classification including reputation

In our proposed method when a node with a lower reputation wants to send a packet to a node with a higher reputation would not be able to send the packet and this attempt would lower the reputation of the sender node. In the next case, when a node with a higher reputation wants to send a packet to a node with a lower reputation would be able to send a packet and by this process, it would increase its reputation.

This system has two-fold benefits. The first benefit is to improve classification goodness. The second benefit is to improve the reputation of good nodes in the overall network and other hand lower the reputation of bad nodes.

In the given three graphs we compare three cases with different node threshold values (T_m). These graphs show the dynamic nodes' reputation assignment 'Rn*' based on the current reputation assignment 'Rn'. These graphs are plotted with Example nodes with increasing reputation vs Reputation score (RP). In the graph (Fig. 6) we take the threshold value (T_m) 0.25 which shows node acceptance is highest with 'Green markers' than the other two graphs. In the graph (Fig. 7) we take the threshold value (T_m) 0.50 which shows node acceptance is higher with 'Green markers' than the last graph (Fig. 8) (where threshold value 0.75) but lower than the graph (Fig. 6) (where threshold value 0.25). In the graph (Fig. 8) we take the threshold value (T_m) 0.75 which shows node acceptance is lowest with 'Green markers' than the other two graphs.

The reputation formula is given below in Eq. 40:

$$\begin{aligned} R_n^* &= 0\, \mathit{if}\, R_n + \left(R_n - T_m\right) * T_m \le 0 \\ &= 1\, \mathit{if}\, R_n + \left(R_n - T_m\right) * T_m \ge 1 \\ &= R_n + \left(R_n - T_m\right) * T_m, \mathit{otherwise} \end{aligned} \tag{40}$$

When the number of acceptances of a node is increased then the reputation of that node will also increase and vice versa, which has been depicted in these given graphs. In the Fig. 6, we consider the threshold value is 0.25. This has been shown that the upper triangle of this point made by the lines R_n(Green line) and R_n(Blue lines) and shaded by green lines (which depicts reputation has been increased) is bigger than the triangle below of this point made by the same lines and shaded by red color (which depicts reputation has been decreased). In the Fig. 7, we consider the threshold value is 0.50. Here the number of acceptances of a node is decreased from before. In this case, the upper triangle and lower triangle are the same in size.

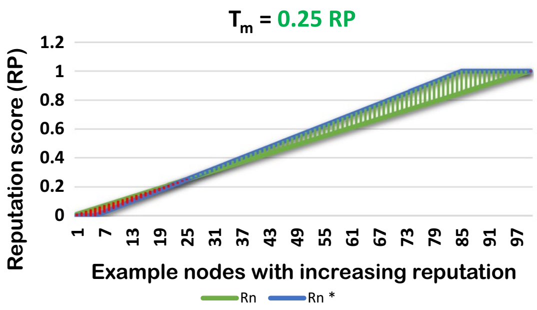


Fig. 6 Impact of node's behavior on its reputation for threshold:0.25

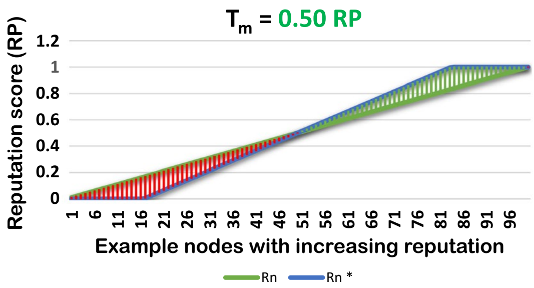


Fig. 7 Impact of node's behavior on its reputation for threshold:0.50

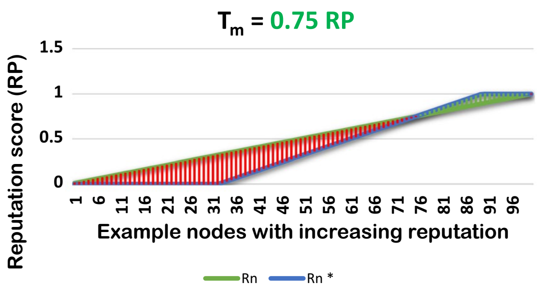


Fig. 8 Impact of node's behavior on its reputation for threshold:0.75

In the Fig. 8, we consider the threshold value is 0.75. Here the number of acceptances of a node is the lowest among these three cases. In this case, upper triangle is smaller than the lower triangle means that acceptance of a node lowest in this case among the three cases we consider.

5 Results and Discussion

5.1 Result Objectives

The following are the objectives of the results section:

1. Identification of the evaluation metrics used for comparison of the algorithms.

2. To find variation in the model performances for general algorithms i.e., SVM and DNN versus the integration of reputation schemes with those algorithms i.e., RSVM and RDNN.
3. A statistical analysis of model verification using ANOVA.
4. To compare the proposed models with the current state-of-the-art techniques.

5.2 Evaluation Metrics

5.2.1 Confusion Matrix

A classification prediction outcome summary is known as a confusion matrix. Confusion matrices are used to depict the counts of predicted and actual values. The confusion matrix serves as the foundation for all other measurements like *precision*, *recall*, *F1 score,* and *accuracy*.

True Negative (TN): The number of correctly identified negative cases.

True Positive (TP): The number of correctly classified positive cases.

False Positive (FP): The number of genuine negative examples that have been misclassified as positive.

False Negative (FN): The number of true positive examples categorized as negative.

5.2.2 Precision

It's the number of correct positive outcomes divided by the classifier's anticipated positive results. Equation 41 is provided that is used to calculate Precision.

$$Precision(P) = \frac{TP}{TP + FP} \tag{41}$$

In our proposed scheme no attacker is allowed to intrude into the system. So our objective is to reduce the false negative (FN), which is intended to increase false positive (FP). So, precision should be decreased.

5.2.3 Recall

It is calculated by dividing the number of accurate positive findings by the total number of relevant samples (all samples that should have been identified as positive). Equation 42 is used to calculate Recall.

$$Recall(R) = \frac{TP}{TP + FN} \tag{42}$$

In our proposed scheme we are trying to decrease the false negative (FN). So, we must increase the recall.

5.2.4 F1 Score

The Harmonic mean of precision and recall is used to get the F1 Score. F1 Score is in the [0, 1] range. It indicates both the precision and the robustness of the classifier. F1 Score attempts to strike a compromise between recall and accuracy. Equation 43 is provided which is used to calculate F1 Score.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{43}$$

Since in our proposed scheme precision should be lesser and recall should be higher and the F1 score is the harmonic mean of both so this is an important metric to measure.

5.2.5 Accuracy

This metric measures the proportion of accurate predictions to all input samples. Equation 44 is provided that is used to calculate Accuracy.

$$Accuracy(A) = \frac{TN + TP}{TN + FP + FN + TP} \tag{44}$$

5.3 Model Variance Evaluation

The Bar chart of Fig. 9 depicted that precision is higher than recall in SVM and DNN. But when we introduce the reputation of nodes in our system i.e., in RSVM and RDNN the value of recall has become higher than precision establishing our objective. The line chart in Fig. 10 depicted that with the increment of threshold, precision also increased. The precision of RDNN is always higher than that of RSVM by 1.5% on average. The Line chart of Fig. 11 also depicted that with the increment of threshold, recall also increased. The recall of RDNN is always higher than RSVM by 0.96% on average. The Line chart of Fig. 12 depicted the F1 score, which is the harmonic mean of precision and recall. The F1 score of RDNN is always higher than that of RSVM by 1.26% on average. The Line chart of Fig. 13 depicted Accuracy. Initially, the accuracy of RSVM is higher than RDNN, but with the increment of the threshold, the accuracy of RDNN would become higher. We can conclude that accuracy is not so consistent for this domain.

5.4 ANOVA Analysis

It is important to test whether the error in the proposed scheme is statistically significant or not. Hence, an ANOVA test is performed to test the statistical significance of the proposed reputation scheme.

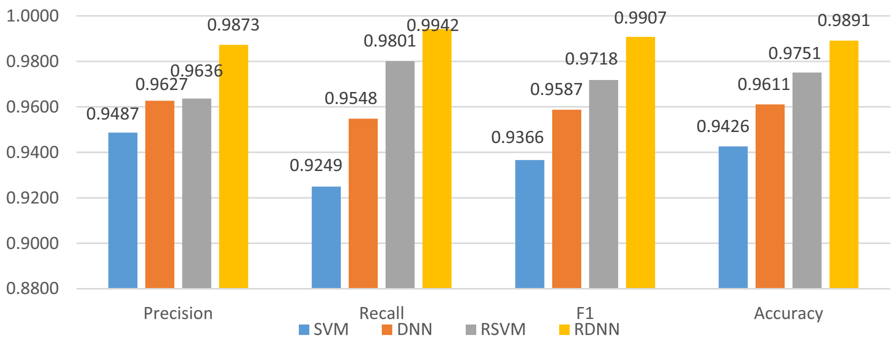


Fig. 9 Different evaluation metrics comparison of four classifiers

Fig. 10 Line chart of Precision of RSVM and RDNN

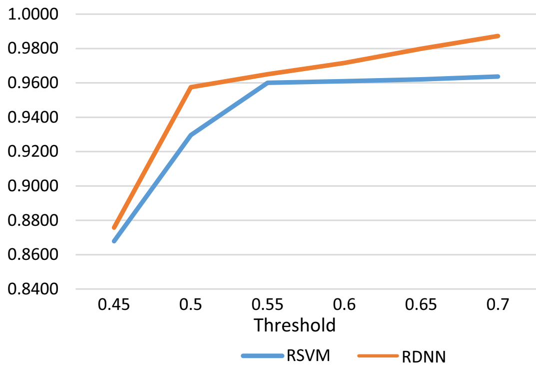
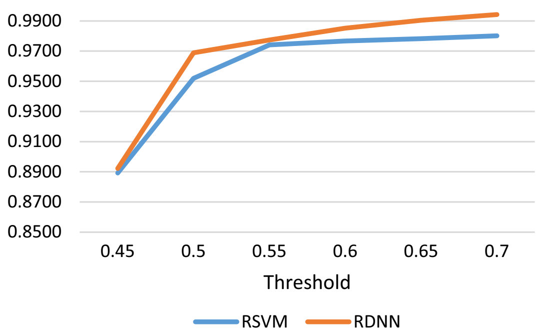


Fig. 11 Line chart of Recall of RSVM and RDNN



We have considered that the effect is fixed for all the treatments with 0.5 level of significance. We assumed a smaller effect of size 0.1 with *F-test* statistics. Further, it should be noted that the outliers are also included in the entire test as per the previous experiments. The hypothesis to be tested is that there is not a major variation before and after the introduction of the reputation scheme based on the data given in the Table 3.

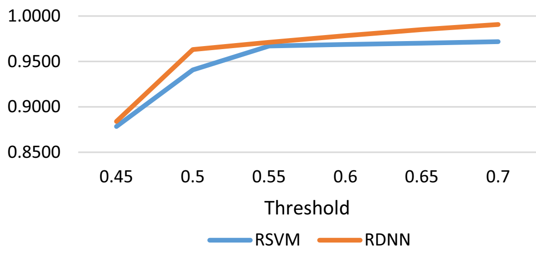


Fig. 12 Line chart of F1 Score of RSVM and RDNN

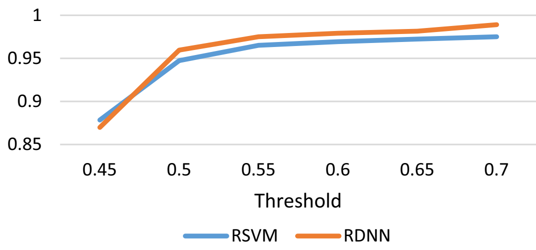


Fig. 13 Line chart of Accuracy of RSVM and RDNN

Table 3 ANOVA test results analysis

Classifier/Metrics	Precision	Recall
Without reputation	0.9487	0.9249
	0.9627	0.9548
With reputation	0.9636	0.9801
	0.9873	0.9942

As per the analysis presented in the Table 4 the H_0 hypothesis for both factors cannot be rejected since the p-value > α. This means that the error difference between the sample averages with and without reputation is not big enough to be statistically significant. The p-value for factor-A equals 0.95 (*P(x ≤ 0.0048) = 0.054*) and for factor-b equals 1 (P(x ≤ 0.000018) = 0.0033). A larger p-value means there is less chance of rejection of H_0. The F-test statistics F_A equals 0.0048, which is in the 95% region of acceptance: [− ∞, 5.32] and F_B equals 0.000018, which is in the 95% region of acceptance: [− ∞, 5.32]. After this test, we can conclude that the reputation scheme can provide a better recall score without changing the overall error which is significant enough for performance degradation.

5.5 Comparison with Existing Approaches

We consider accuracy when evaluating a model, but we are more concerned with how resilient it is, how it will perform on diverse datasets, and how much flexibility it provides. Without a question, accuracy is a crucial statistic to evaluate, but it does not always provide a whole picture. Accuracy is a useful metric for evaluating a model's performance in a problem set. This can also be used to rank and compare

Table 4 Analysis of H_0 hypothesis

Source	DF	Sum of square (SS)	Mean square (MS)	F statistic (df1, df2)	P-value
Metrics (A)	1	0.0015	0.0015	0.0048 (1,8)	0.95
Classifier (B)	1	0.0000057	0.0000057	0.000018 (1,8)	1
Interaction AB	1	0.00025	0.00025	0.00082 (1,8)	0.98
Error	8	2.48	0.31		
Total	11	2.48	0.23		

different models. Some of the metrics help explain how the model captures the problem and interprets the data. The proportion of correct predictions to all input samples is known as accuracy. From here, we cannot deduct false positives or false negatives Table 5.

A false positive is an outcome when a model incorrectly predicts the positive class and a false negative is an outcome when the model incorrectly predicts the negative class. A false positive in a real-life scenario can prove to be quite damaging. In Intrusion Detection System, a false negative case happens when an action is classified as normal, even though it is malicious. A confusion matrix is a specific table layout of dimensions 2×2, that helps us visualize the performance of any Machine Learning algorithm. It helps us to calculate values like Precision, Recall, Specificity, F-1 score, and ROC-AUC curve. These metrics, in addition to accuracy, let us understand a model's performance even better. An unknown dataset in machine learning doesn't matter when we train a model and if the model is a generalized model or a good model. Because a generalized model works on unknown data. Then which dataset we would select for our experiment depends on our requirements and

Table 5 Comparisons of accuracy

State of the art techniques	Used dataset	Accuracy
RSVM	Kitsune network attack dataset	97.51%
RDNN	Kitsune network attack dataset	98.91%
Naïve Bayes [41]	CCIDS2017	75.31%
SVM [41]		99.68%
Random tree [22]	ISCX-SlowDDos-2016	99.95%
k-NN [42]	UNSW-NB15	92%
	NSL-KDD	96%
Neural network [23]	UNB-ISCX	98%
Deep learning neural network [24]	KDD Cup	97.10%
REPTree + SVM [25]	CTU-13	98.40%
RNN neural network [43]	CTU-13	98.39%
k-means clustering + Naïve Bayes [26]	UNB-ISCX	99%
Random forest [27]	CIC DoS-2017	94.41%
SVM [27]		93.10%

source verification. We selected the Kitsune Network Attack (KNA) dataset for our experiment since the number of instances of these datasets is 27170754 and the number of attributes is 115 and many attributes are required for MANET attacks in the network. We try to make the model generalized so that it not only works on KNA but works for any unknown dataset.

6 Conclusions

MANET's advantages like its dynamic and decentralized nature also bring a lot of disadvantages when compared to any wired network technology. We identified the challenges that state-of-the-art machine learning models face when classifying a real DoS attack versus a false classification due to a network error. A reputation-based approach is proposed assuming that a node's history plays a very important role in determining whether the node is an attacker or not. This proposed approach in a lab environment simulation shows that it can improve the classification accuracy of existing machine learning models to a large extent. The reputation-based method can stop classifiers from discarding a node directly when a node has a good reputation and vice versa. The minimum increment in accuracy is 2.8%, which increases to 3.25% for other models. Further, the model recall is increased by a mean of 1% for all the tested models, which is a significant improvement considering the cost associated with the false classifications. The only limitation identified by the proposed approach is the cold start issue; as the dynamic nature of the MANETs does not allow a centralized system to handle the reputation system.

Author Contributions SJ, NS: wrote the main manuscript text and SD, PC: prepared all figures. All authors reviewed the manuscript.

Funding Not applicable.

Data Availability We have used Kitsune Network Attack Dataset [33] for our experiment. Kitsune is an online, unsupervised, and efficient ANN-based network intrusion detection system (NIDS). https://archive.ics.uci.edu/ml/datasets/Kitsune+Network+Attack+Dataset

Declarations

Competing interests As a part of my research, my domain of interest is Mobile Adhoc Network (MANET), and Artificial Intelligence (AI); this is my continuous learning process and there is no financial relationship with other people or organizations.

Ethical Approval Not applicable.

References

1. S. Yi, P. Naldurg and R. Kravets, *2001*. "Security-Aware Ad hoc Routing for Wireless Networks," in *The Second ACM Symposium on Mobile Ad Hoc Networking & Computing, MobiHoc,*
2. L. Yi, Y. Zhong and B. Bhargava, 2003 "Packet Loss in Mobile Ada Hoc Networks," Department of Computer Science Technical Reports. Paper 1558.
3. B. Sun, Y. Guan, J. Chen and U. W. Pooch, 2003 "Detecting black-hole attack in mobile ad hoc networks," in 5th European Personal Mobile Communications Conference, Glasgow, UK,.

4. M. Al-Shurman, S. Yoo and S. Park, 2004 "Black hole attack in mobile ad hoc networks," in *ACM-SE 42:* Proceedings of the 42nd annual Southeast regional conference,.
5. V. Gupta, S. Krishnamurthy and M. Faloutsos , 2002 "DENIAL OF SERVICE ATTACKS AT THE MAC LAYER IN WIRELESS AD HOC NETWORKS," in *MILCOM 2002.* Proceedings, Anaheim, CA, USA,.
6. Xiao, P., Qu, W., Qi, H., Li, Z.: Detecting DDoS attacks against data center with correlation analysis. Comput. Commun. **67**(0140–3664), 66–74 (2015)
7. P. K. Agrawal, B. B. Gupta and S. Jain, 2011 "SVM Based Scheme for Predicting Number of Zombies in a DDoS Attack," in European Intelligence and Security Informatics Conference, Athens, Greece,
8. S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix and P. Hakimian, "Detecting P2P botnets through network behavior analysis and machine learning," in Ninth Annual International Conference on Privacy, Security and Trust, Montreal, QC, Canada, 2011.
9. S. Sambang and L. Gondi , "A Machine Learning Approach for DDoS (Distributed Denial of Service) Attack Detection Using Multiple Linear Regression," in The 14th International Conference on Interdisciplinarity in Engineering—INTER-ENG 2020, Târgu Mureș, Romania.
10. Fadil, A., Riadi, I., Aji, S.: Review of detection DDOS attack detection using naive bayes classifier for network forensics. Bulletin of Electrical Eng. and Infor. **6**(2), 140–148 (2017)
11. Wu, Y.-C., Tseng, H.-R., Yang, W., Jan, R.-H.: DDoS detection and traceback with decision tree and grey relational analysis. Int. J. of Ad Hoc and Ubiquitous Comput. **7**(2), 121 (2011)
12. M. Suresh and R. Anitha, 2011 “Evaluating machine learning algorithms for detecting DDoS attacks,” in *Springer*, Berlin, Heidelberg,.
13. S. Akram, I. Zubair and M. H. Islam, "Fully distributed dynamically configurable firewall to resist DOS attacks in MANET," in First International Conference on Networked Digital Technologies, Ostrava, Czech Republic, 2009.
14. Sahu, L., Sinha, C.: A Cooperative approach for understanding behavior of intrusion detection system in mobile ad hoc networks. Int. J. Comput. Sci. Mobile Appl. - Ijcsma **1**(1), 24–30 (2013)
15. H. X. Tan and W. Seah, 2005 "Framework for Statistical Filtering Against DDoS Attacks in MANETs," in Second International Conference on Embedded Software and Systems (ICESS’05), Xi’an, China,,.
16. S. Banerjee , R. Nandi, R. Dey and H. N. Saha, 2015 "A review on different Intrusion Detection Systems for MANET and its Vulnerabilities," in *International Conference and Workshop on Computing and Communication (IEMCON)*, Vancouver, BC, Canada,.
17. Y. A. Huang and W. Lee, "Hotspot-based Traceback for Mobile Ad Hoc Networks," in WiSe ’05: Proceedings of the 4th ACM workshop on Wireless security, Cologne, Germany, 2005.
18. J. Ioannidis, "DoS Pushback. Encyclopedia of Cryptography and Security," *Springer,* 2011.
19. Denko, M.K.: “Detection and prevention of denial of service (DoS) attacks in mobile ad hoc networks using reputation-based incentive scheme,.” J. Syst. Cybernet and Inform. **3**, 4 (2005)
20. Wenjia, L., Parker, J., Joshi, A.: Security through collaboration and trust in MANETS. Mobile Net and Appl. **17**, 342–352 (2012)
21. X. Wu and D. K. Y. Yau,, 2007 "Mitigating denial-of-service attacks in MANET by distributed packet filtering: A game-theoretic approach," in *ASIACCS ’07:* Proceedings of the 2nd ACM symposium on Information, computer and communications security.
22. A. B. Dehkordi and M. Soltanaghaei,, 2020 "The DDoS attacks detection through machine learning and statistical methods in SDN," The Journal of Supercomputing © Springer Science+Business Media, LLC, part of Springer Nature,.
23. Saied, A., Overill, R.E., Radzik, T.: Detection of known and unknown DDoS attacks using artificial neural networks. Neurocomputing **172**, 385–393 (2015)
24. Sumathi, S., Karthikeyan, N.: Detection of distributed denial of service using deep learning neural networks. J. of Ambient Intell. and Humanized Computin (2020). https://doi.org/10.1007/s12652-020-02144-2
25. Kalaivani, P., Vijaya, M.: Mining based detection of botnet traffic in network flow. Int. J. of Comput. Sci. Inf. Technol Secur. **6**, 535–540 (2016)
26. W. Yassin, N. I. Udzir, Z. Muda and M. N. Sulaiman, 2013 "Anomaly-based intrusion detection through k-means clustering and Naives Bayes classification," in 4th International Conference on Computing and Informatics, ICOCI, Sarawak, Malaysia,.
27. Pérez-D˝AzValdovinosChooZhu, J.A.I.A.K.R.D.: A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning. IEEE Access **8**, 155859–155872 (2020)
28. Pérez, F., Granger, B.E.: IPython: a system for interactive scientific computing. Comput. Sci. and Eng., publisher IEEE Comput. Soc. **9**, 21–29 (2007)

29. M. Abadi, A. Agarwal and P. Barham, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. [Online]. Available: https://www.tensorflow.org/
30. Harris, C.R., Millman, K.J., van der Walt, S.J.: Array programming with NumPy. Nature **585**, 357–362 (2020)
31. Pedregosa, F., et al.: Scikit-learn: machine learning in python. J. machine learn. res. **12**, 2825–2830 (2011)
32. Hunter, J.D.: Matplotlib: A 2D graphics environment. Comput. Sci. Eng. **9**, 90–95 (2007)
33. Y. Mirsky, T. Doitshman, Y. Elovici and A. Shabtai, "Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection," in Network and Distributed System Security Symposium 2018 (NDSS'18).
34. M. Patel and S. Sharma, "Detection of malicious attack in MANET a behavioral approach," in IEEE International Advance Computing Conference, IACC, Ghaziabad, India, 2013.
35. T. Poongothai and K. Duraiswamy, 2014 "Intrusion detection in mobile AdHoc networks using machine learning approach," in International Conference on Information Communication and Embedded Systems *(ICICES)*, Chennai, India
36. S. Laqtib, K. E. Yassini and M. L. Hasnaoui, 2019 "A Deep Learning Methods for Intrusion Detection Systems based Machine Learning in MANET," in *SCA '19:* Proceedings of the 4th International Conference on Smart City Applications, CASABLANCA, Morocco.
37. Catak, F.O., Mustacoglu, A.F.: Distributed denial of service attack detection using autoencoder and deep neural networks. J. Intell. Fuzzy Syst. **37**, 1064–1246 (2019)
38. C. Xu, J. Shen and X. Du, 2021 "Low-rate DoS attack detection method based on hybrid deep neural networks," *Journal of Information Security and Applications, Elsevier,* pp. 2214–2126,.
39. Shieh, C.S., Lin, W.W., Nguyen, T.T., Chen, C.H.: Detection of unknown DDoS attacks with deep learning and gaussian mixture model. Appl. Sci. (2021). https://doi.org/10.3390/app11115213
40. Y. Li and Y. Lu, "LSTM-BA: DDoS Detection Approach Combining LSTM and Bayes," in Seventh International Conference on Advanced Cloud and Big Data (CBD), IEEE, Suzhou, China, 2019.
41. Rimal, A.N., Praveen, R.: DDOS attack detection using machine learning. J. Emerging Technol. and Innovative Res. **7**(6), 185–188 (2020)
42. A. Krivchenkov, B. Misnevs and A. Grakovski, 2020, "Using Machine Learning for DoS Attacks Diagnostics," in *RelStat 2020:* Reliability and Statistics in Transportation and Communication, Springer Nature Switzerland AG. Part of Springer Nature.
43. A. Bansal and S. Mahapatra, 2017 "A comparative analysis of machine learning techniques for botnet detection," in *SIN '17:* Proceedings of the 10th International Conference on Security of Information and Networks.

Subrata Joardar is affiliated as Research Scholar to Computer Science and Engineering, National Institute of Technology, Durgapur. Currently providing services as Senior Lecturer (Selection Grade-II) in Computer Sc. & Technology at Siliguri Govt. Polytechnic. His scientific research interests include Security in Mobile Ad-hoc Network, Wireless Sensor Networks, Expert Systems and Deep Learning.

Nilanjan Sinhababu is currently pursuing Ph.D. degree from Centre for Computational & Data Sciences, Indian Institute of Technology Kharagpur, India. He recieved his Master of Science (by research) degree from Subir Chowdhury School of Quality and Reliability, Indian Institute of Technology Kharagpur. He received his B.Tech. degree in Computer Science & Engineering from West Bengal Technical University, India. His research interests include Data Analytics, Machine Learning, Crowd Computing, Expert Systems and Deep Learning.

Soumyodeep Dey has received his B.Tech. degree in Computer Science & Engineering from the National Institute of Technology Durgapur (NIT Durgapur). He is currently pursuing his Master's in Computer Science and Engineering from the International Institute of Technology Hyderabad (IIIT HYDERABAD). His research interests include Natural Language Processing, Machine Learning and Recommendation systems.

Prasenjit Choudhury is affiliated to Computer Science and Engineering, National Institute of Technology, Durgapur. Currently providing services as Associate Professor. He has published numerous publications in various national and international peer-reviewed journals and presented scientific papers across the world. His clinical and scientific research interests include Security and Service Management in Mobile Ad-hoc Network, Mobile Crowd Computing, Data Science.