Hindawi
Journal of Computer Networks and Communications
Volume 2019, Article ID 4683982, 14 pages
https://doi.org/10.1155/2019/4683982

Research Article

Detection and Prevention of Man-in-the-Middle Spoofing Attacks in MANETs Using Predictive Techniques in Artificial Neural Networks (ANN)

Robert A. Sowah, Kwadwo B. Ofori-Amanfo, Godfrey A. Mills, and Koudjo M. Koumadi

Department of Computer Engineering, University of Ghana, PMB 25, Legon, Accra, Ghana

Correspondence should be addressed to Robert A. Sowah; rasowah@ug.edu.gh

Received 27 June 2018; Revised 21 November 2018; Accepted 13 December 2018; Published 20 January 2019

Academic Editor: Zhiyong Xu

A Mobile Ad-Hoc Network (MANET) is a convenient wireless infrastructure which presents many advantages in network settings. With Mobile Ad-Hoc Network, there are many challenges. These networks are more susceptible to attacks such as black hole and man-in-the-middle (MITM) than their corresponding wired networks. This is due to the decentralized nature of their overall architecture. In this paper, ANN classification methods in intrusion detection for MANETs were developed and used with NS2 simulation platform for attack detection, identification, blacklisting, and node reconfiguration for control of nodes attacked. The ANN classification algorithm for intrusion detection was evaluated using several metrics. The performance of the ANN as a predictive technique for attack detection, isolation, and reconfiguration was measured on a dataset with network-varied traffic conditions and mobility patterns for multiple attacks. With a final detection rate of 88.235%, this work not only offered a productive and less expensive way to perform MITM attacks on simulation platforms but also identified time as a crucial factor in determining such attacks as well as isolating nodes and reconfiguring the network under attack. This work is intended to be an opening for future malicious software time signature creation, identification, isolation, and reconfiguration to supplement existing Intrusion Detection Systems (IDSs).

1. Introduction

Computer security is one of the areas in computer technology which have attracted much interest from many security professionals and "lay" persons. This field was necessitated by previously known and newly developing techniques which afford attackers the means to launch sophisticated attacks, giving them access to resources on networks and compromising those networks in the process. Notable among such established techniques are distributed denial of service (DDOS) attacks, man-in-the-middle (MITM) spoofing attacks, and session hijacking [1]. With man-in-the-middle spoofing attacks, the focus of this research, a third party—the attacker in this case—basically inserts himself between two parties or devices in stealth mode in such a way that all packets between those two legitimate parties are routed through him. This is quite malicious because the attacker can then alter the information in the packets, potentially sending falsified data to either party [2].

Regarding categorization, MANETs could be seen as VANETs, internet-based mobile ad hoc networks (MANETs), or military-based MANETs. These broad categorizations imply that MANETs have comprehensive operational capabilities. However, it is their diversity that makes such networks very susceptible to the aforementioned attacks [3–5].

It is interesting to note that attacks are not just limited to particular devices. Mobile-targeted attacks can be performed against small to very high targets and across multiple platforms [6, 7]. Even the various attacks could be broken down based on which software application they have been tuned to violate. The man-in-the-middle attack, for instance, has a slight variant called the man-in-the-browser attack which is specific to browser-based applications and services

[8]. It is aimed at intercepting communications between several clients on browser platforms. Numerous variants of session hijack attacks and buffer overflow attacks exist, with such attacks, unlike in the past, being automatable via software tools. Wireshark, Nmap, and Tcpdump are among the variety of tools available to today's hackers, and these have even given rise to a new breed of hackers called click-kiddies who, in comparison to the earlier breed of hackers, have relatively little or no programming experience [9]. These are also capable of initiating sophisticated attacks against prime targets. With the sharp growth in the processing power of hardware, as well as the exponential development of software tools and programming languages, the amount of power available to a single user in a cyber-network has never been more significant than in today's world. It is therefore no surprise that numerous initiatives and investments are being made to protect wired and wireless networks from prying eyes. History shows that technological development has for centuries been military driven. It is therefore ironic to see the military turn to civilian experts in cyber defense [9]. Technological development represents possibly the one single area where civilians can compete on a par with the military; thus, it has over the last couple of years offered numerous challenges to both worlds.

Machine learning algorithms such as support vector machines, neural networks, and others hold promise for learning the complex behavioural patterns needed for cyber defense. Artificial neural networks (ANN) modelled after the human brain functionalities have great potential. In recent times, interest has been rekindled in ANN due to new knowledge garnered from psychology and human behaviour. This has led to new frontiers and possibilities in deep learning (a set of machine learning algorithms targeted at modelling high-level abstractions) [10, 11]. It is against this backdrop that this research has been carried out to use ANN techniques to solve the problem of the man-in-the-middle attack.

This paper is organized into sections. Section 2 gives an overview of some existing works that have already been carried out on MANET security. The problem statement and its formulation are presented in Section 3. Captured under Section 4 is the detection model design and development with its flowcharts. Section 5 presents the model implementation and testing done on the developed modules and their integration. Results and discussions on the experimental setup and simulations are in Section 6. Finally, Section 7 presents the conclusions that were arrived at as well as recommendations for future work.

2. Related Works

The central problem inherent in the operation of MANETs is the difficulty in detecting and counteracting man-in-the-middle spoofing attacks. These networks are quite unsecured since they offer opportunities by which hackers can get and exploit resources on them [12–15]. This has spawned several researches into maintaining their security, some of which have been indicated in the previous section. This work presents introduced neural network techniques for detecting fraudulent nodes involved in man-in-the-middle spoofing attacks which constitutes one of the most challenging types to detect and prevent.

There have been many works in the field of network security each adopting unique techniques to accomplish its purpose. There have been works such as "Secure data protocol for distributed wireless sensor" [16], "Networks, secure program execution in wireless sensor networks," and "Intrusion detection model in MANETs using ANNs and ANFIS" [11]. These highlight the broad spectrum of researches carried out and ongoing, from intrusion detection to tracing and prevention, cryptography, and network monitoring. It is against the background of such good works that this research sought alternative and improved techniques by adopting neural networks to investigate and solve problems relating to man-in-the-middle spoofing attacks on and across MANETs. Such methods offer more flexible advantages which enable detection and manual prevention of such attacks, as well as functionalities that will allow administrators and security services to trace and apprehend offenders. It is worth noting that man-in-the-middle attack is one of the most difficult-to-detect kinds of attacks. That is where intelligent techniques such as artificial neural networks come in since they offer the ability for the network to learn and generalize to new scenarios [10, 11].

Detection of attacks in MANETs is a growing field in networks security. Although there have been several approaches at preventing both proactive (attacks initiated without any prior information on the victim's system) and reactive (attacks initiated based on an initial response of the system under attack to a previous attack or stimulus such as SQL-injections), attacks as captured in Kurosawa et al. [17] as well as Reidemeister and Böhm [18], none of these papers provides an approach for the possible simulation and remedying of a man-in-the-middle attack on a simulating platform—the NS2 environment. Classification algorithms have been quite widespread in their use of intrusion detection systems but our approach adopted artificial neural networks.

In [19], the authors introduce the concept of a formal validation methodology for MANET routing protocols based on nodes' self-similarity. The aim was to fill the gap left by simulation or emulation tests without having to perform the probable and feasible, otherwise costly alternative of actual testing. It sought to apply a conformance testing approach neglected in the aforementioned approaches, by using the Dynamic Source Routing Algorithm (DSR). It highlighted the disadvantage of using simulation tests as their inability to completely mimic real-world scenarios. However, the paper also highlighted the problem of formal method-based approaches as being their inability to consider the inherent MANET protocol characteristics. Thus, in this work, a trade-off was made of the latter, adopting a simulation-based approach in the process.

On the authentication schemes that have been investigated, Maag et al. [19] provides insights into various schemes, providing an alternative way of offering authentication called HEAP—an HMAC-based algorithm which

utilizes two keys. The target of the new schemes was to prevent popular attacks such as DDoS and man-in-the-middle attacks. Although this approach offered the advantages of lower memory requirement in comparison to other existing schemes such as TESLA and LHAP, with limited CPU and bandwidth overhead, it constrained itself to detecting attacks from outsider nodes.

Insider attack detection was apparently left to the installed IDS. Thus, in a scenario where an IDS was not installed, such attacks could go undetected. The approach taken during this research did not have that constraint, having been envisaged to detect attacks from both outsiders and insiders.

There are several attacks possible on MANETs as enumerated in [3, 20] with their corresponding techniques of defending beside them. The issue of performance was specifically touched upon in this work for both MANETs with Internet access—the greatest source of most network-based attacks in today's world and for standalone MANETs. Suggestions were made at the end for the use of a framework that utilizes minimal public key cryptography interaction to offer security. The conclusion was that an overelaborate use of such key infrastructure overloaded the network and reduced performance. That also informed our using of using artificial neural networks with learning capabilities and good generalizability.

Abdalla et al. [21] showed the possibilities associated with a trio-ID message-based approach to perform attack detection and node isolation. The major advantage that this approach offered was the shift in the attack detection responsibility from all the interacting nodes to just the source node. A cumulative effect could be a conservation of network power from the reduction in computation requirements. However, the main disadvantage noted was that smaller attacks running for comparatively shorter time spans could go undetected. Such attacks needed to be run for a longer time to trigger the level of detectability that is so obvious in larger attacks. That is where a trained neural network has the envisaged advantage of being able to notice minute changes after training and retraining. The published literature that presented a detection and prevention approach is in [4, 17, 22, 23]. It differs from [21] not only in the approach used but also in the attack type which was considered. While Abdalla et al. [21] focused on packet dropping nodes, Chen et al. [22] was more interested in IP-spoofed initiated DDoS attacks, picking as its chosen method a technique that essentially "coloured" the path a packet traverses, to enable easier tracking of a promiscuous source for elimination from the network. Presenting an advantage of a lower deployment cost as opposed to other packet marking schemes, it helped eliminate illegitimate nodes by marking legitimate ones, a different approach to the Packet Identification (PI) mechanism it sought to compete against. It offered a 70% acceptance ratio realization when there were only 20% of routers participating in the scheme. That was far better than the 60% acceptance ratio offered by PI, achieved with all routers participating. The parameters used in developing the newer technique presented in the paper were determined through heuristics. That is informative in presenting the possibility of adopting such an approach during the feature extraction and machine learning phase of any research for detecting spoofing attacks.

There were many possibilities for network attack creation, namely, (1) a real attack (which would have been resource utilization-intensive and quite expensive in its execution) and (2) a simulated attack via the use of an emulator or performing a computer simulation. After reviewing the results given in [24], it was realized that a simulation-based approach was more appropriate due to the costs involved. This paper offered insightful thoughts and advice on how to perform black hole attacks using the NS2 simulation software. By making use of the RREQ, RREP packets, it could mimic scenarios of a black hole attack. Analysis of the analogies made was helpful in arriving at a similar but considerably different approach towards simulating a MITM attack on an NS2 platform. Instead of dropping packets as given in [24], it was chosen to induce a slight delay, as was hypothesized by this research to be expected in an MITM attack. Several factors had to be considered before coming up with a classification algorithm.

Almost every approach used in attack detection has the probable drawback of pulling in some false positives as attacks. That was highlighted by Mitrokotsa and Dimitrakakis [23]. With a comparative analysis of cost-sensitive classification being done, tuning of hyperparameters was essential as the experimental protocol for the goals of checking the influence of altered cross-validation methods on classification algorithms. The comparative analysis investigated how weighted classification schemes contributed to classification accuracy improvements. This paper offered a conclusion that since several algorithms have security considerations as a focus, it is better to err on the side of caution rather than to "clean up the mess later." The key idea put forward in several prior published research works, which this work affirms and adopts, was that the cost incurred by flagging a false positive was far less in comparison to the damage that would result from an unnoticed attack. Thus, the installation of IDSs does not necessarily imply insulation from all attacks, but rather a minimization of the possibility of an attack, or at least, a reduction in the probability of an attack not being reported. This paper aimed to minimize the number of false positives while increasing the number of detected attacks using neural networks. Overall, there is a lot of literature on MANET attacks but very little of it specifically addresses the problem of man-in-the-middle attacks [4, 11, 15, 25, 26]. This study aims at remedying this situation. Thus, this research specifically focuses on (1) detecting such attacks and correcting first instance cases (MITM attacks scenarios not noticed before) through learning and (2) using the learned experience garnered to prevent future occurrences of such attacks, thereby reducing the cost incurred from unreported attacks or delayed ones. Having surveyed the existing literature and the tremendous effort put into maintaining security on MANETs, this work sought to build upon the foundations already laid. Hence, the research involved the introduction of neural network techniques and IP/MAC address mapping techniques to detect fraudulent nodes. The study focuses on the man-in-the-middle attack mode with the aim of crafting a

viable software model for detecting, recovering from, and preventing future attacks as illustrated in the thesis located in the URL given in [27]. The system monitored the network for detecting and isolating promiscuous nodes and assessing malicious nodes involved in spoofing attacks. Finally, based on the information acquired from observed scenarios, the system could adapt the network to counteract or curtail future exploits.

3. Problem Statement

A man-in-the-middle attack is a computer-based attack in which some third-party masquerades as either party in a two-way communication scenario, to trick one party into thinking that he/she is talking to the other. Under such circumstances, an attacker can eavesdrop on the communications between the two unsuspecting parties to glean information. Such attacks are possible across both wired and wireless infrastructure, with the latter being more susceptible. That is due to the relatively more loosely-defined restrictions on wireless networks. As such, MITM attacks are potent techniques for compromising wireless networks, of which MANETs form a part.

MANETs use two main routing protocols, namely, (1) Ad-Hoc-On-Demand Distance Vectoring (AODV) and (2) Dynamic Source Routing (DSR). There have been several related works to address security in MANETs which have been discussed in Section 2 of this paper. The realization, however, is that there exists no viable dynamic technique for addressing MITM attacks on MANETs using, particularly, the AODV protocol. There exist some static methods, as discussed in Section 2; however, an adaptable method is necessary to address the ever-growing attack threads posed by new and emerging stealth attack threats. The more versatile the technique, the easier it would be to handle unanticipated attack vectors. Artificial neural networks (ANNs), due to their learning and generalizable qualities and owing to their ability to discover information from unintelligible data and infer new information, are more suited to handle such tasks. Therefore, the problem of detecting MITM attacks across MANETs running on the AODV protocol was handled using ANN.

4. MITM Detection Model Design and Development

A 5-node architectural network presented in Figure 1 was modelled for the network attack detection system. In the network, one dedicated node N5 was used as the administrator which monitors the MANET for malicious nodes, dislodges them, and reconfigures the network. The NS2 (*from ns-allinone-2.35.tar.gz package*) simulator together with the NAM animator were used for the model development [28]. The programming languages Perl, C/C++, and Java were used with the combination of scripts and executable codes on Linux Ubuntu 13.10 operating system. For the ANN and other machine learning algorithms, the WEKA machine learning software package and its API provided the essential resources [29].

Conceptually, the Java application can be made to read logged data from any layer of the TCP-IP protocol stack. The features extracted from the logged details can then be used to train the ANN for attack detection. Software vendors or system administrators wishing to use such a code could, as depicted in Figure 2, configure it for single layer logging or multilayer (cross-layer) feature extraction for ANN training.

The flowchart in Figure 3 was initially envisaged with an IDS component being attached to the system to generate log files and help in intrusion detection. The final flowchart was arrived at after the realization that the installation of an IDS could result in extra resource (power and processing time) consumption and that the ultimate neural network system shown in Figure 4 was adequate in detecting and preventing intruding nodes. The model for implementation includes (1) data generation and network modeling, (2) attack formulation, (3) feature extraction, (4) detection system (5) prevention system and recovery, and (6) testing and results with performance metrics.

To implement a system that could perform attacks which could also be monitored for malicious traffic detection and recovery, the methodology was broken down into three steps as depicted in Figure 4 above. L1 represented the stage at which attack simulation and regular network traffic communication were carried out. At L2, attack detection was commenced and the last stage L3 catered for attack recovery and reconfiguration.

5. Detection Model Implementation and Testing

5.1. L1: Attack Simulation Implementation. To implement the attack simulations, the attack model as presented in Figures 1 and 3 was first implemented by creating the AODV_MITM protocol using existing AODV protocol. The AODV_MITM protocol was created by adapting the existing AODV protocol. This was to enable the system to send out attack-type packets during the simulation stage.

To customize the protocol as desired, we altered versions of all aodv files located in the ns-2.35 directory as depicted in Figure 5.

Figure 5 gives a revised representation of the ns-allione-2.35 directory structure with all the salient portions relevant for development and simulations. All the codes relevant for the simulations to be carried out are placed there. The *tcl* folder which has subdirectories such as *lib* and *tests* contains most of the *Otcl* source code necessary for simulations to be carried out. All additional alterations on all customized C++ codes and projects can be put directly in the ns-2.35 folder. That was where the customized version of the *aodv* protocol, *mitmaodv*, installed as part of the simulator, was placed. It was used in the simulation of the man-in-the-middle attack. The files from the original protocol were renamed. The component diagram for the developed *aodvmitm* package is illustrated in Figure 6 below. Apart from the *aodv_packet.h* file, every other file name in that directory was appended with the "*mitm*" string. That was done to ensure that packets could be exchanged between nodes using the native "*aodv*" protocol and the new customized variant "*aodvmitm*."

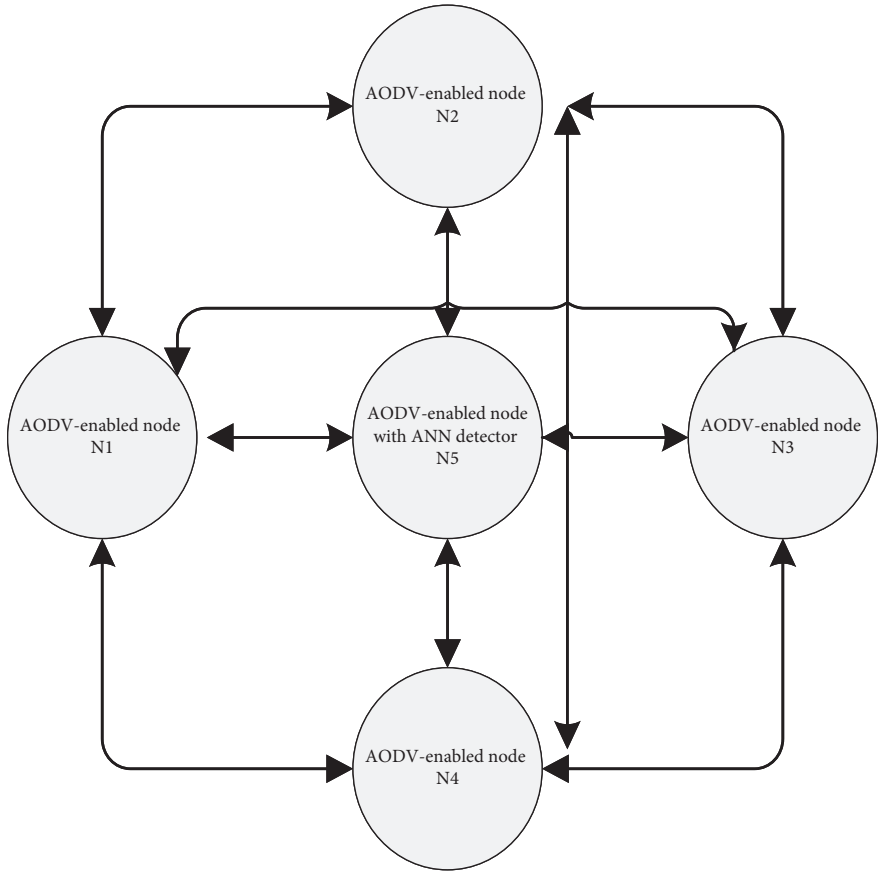

Figure 1: General system architecture of the new working system.

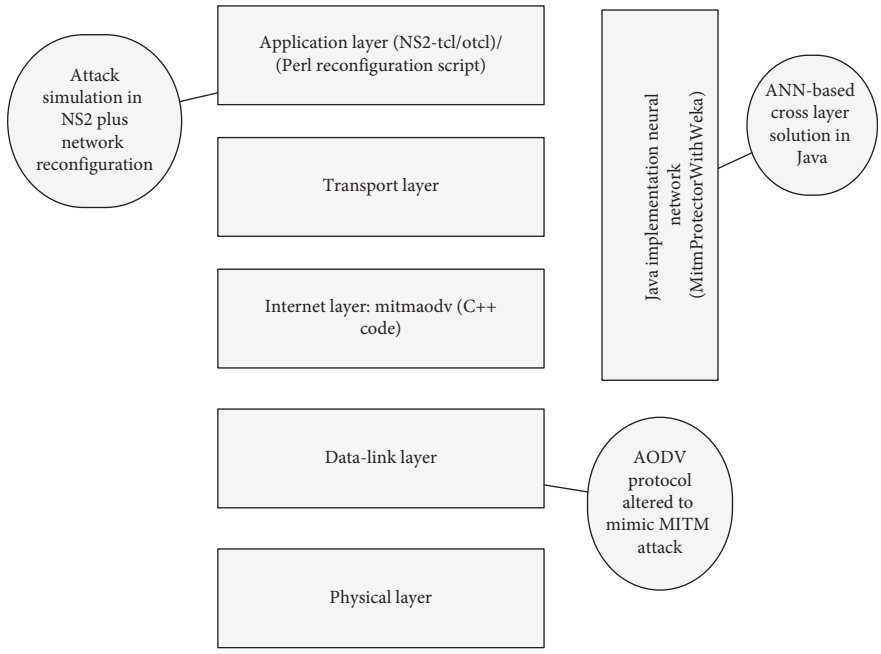

Figure 2: Conceptualization and design.

All classes and structures were renamed in the new protocol's implementation except the ones in the *aodv_packet.h* file [19]. Apart from the addition of the new routing protocol as per requirement, other additions were made in the *ns-packet.tcl* file located in the "*ns-2.35/tcl/lib*" subdirectory. This file is necessary for packet format initialization anytime

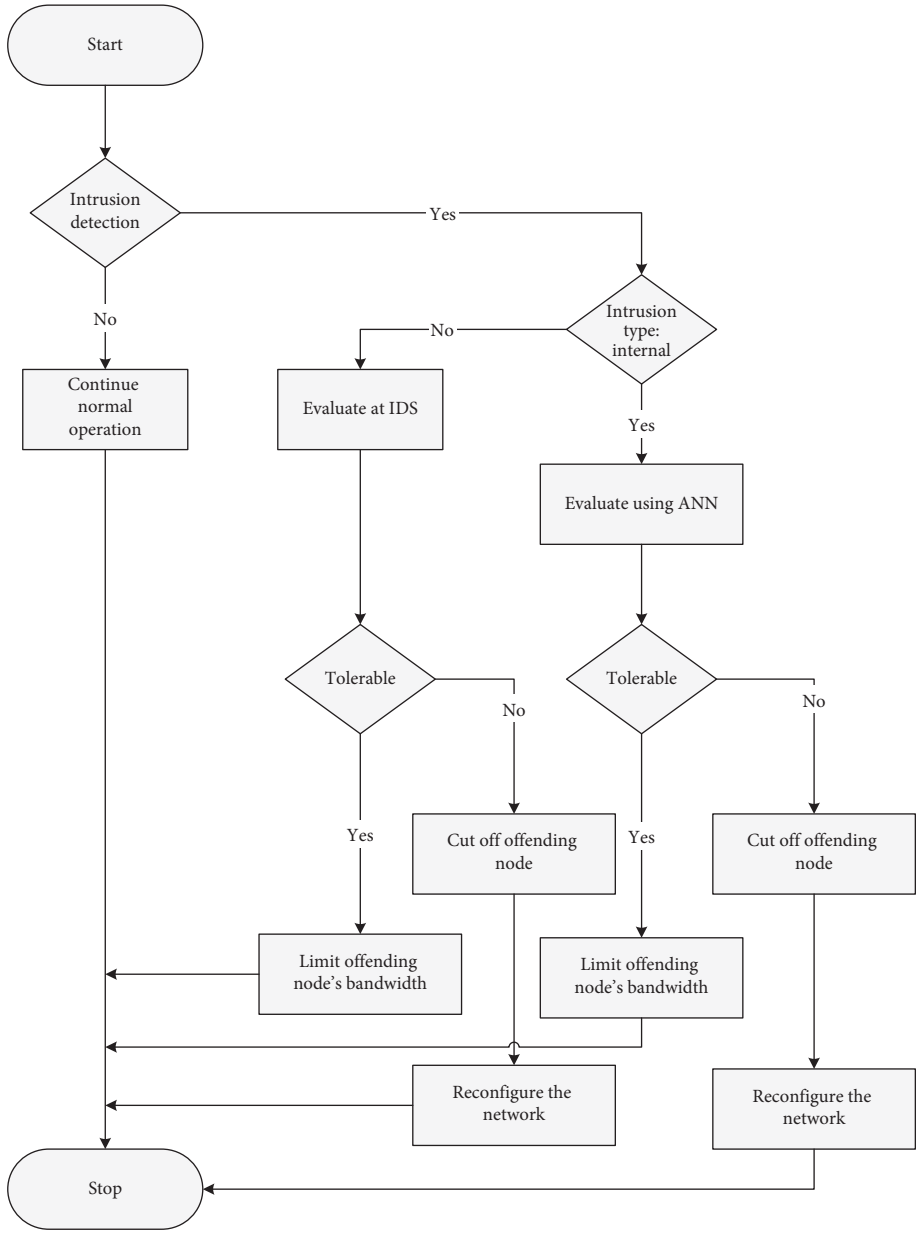

Figure 3: Flowchart diagram of design.

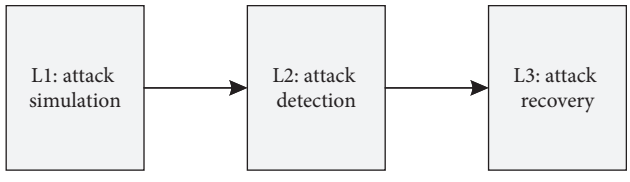

Figure 4: Final condensed flowchart design for implementation.

a simulation is started. Thus, any new packet created needs to be registered in this file as shown in Code Listing 1.

Additionally, several lines in sample Code Listing 2 as well as Code Listing 3 were added to the *ns-lib.tcl* file. This file contains the list of classes and functions that directly mirrors the implementation of classes and function in C++ for NS2's Otcl/C++ linkage.

5.1.1. OTCL: Parameters for sConfiguration. Tcl and Otcl codes were written to carry out the simulations. The node movement was set using the "*setdest*" script -which had as initial parameters the values presented in the first column of Table 1 captured in the "*mitm_attack.tcl*" and "*mitm_attack_reduced.tcl*" files. Each simulation is done as in Figure 7, with the attacking node being the red node and the genuine nodes as those in black. The simulation was done using a minimum of 6 nodes and a maximum of 20 nodes and varied in-between during the stress testing phase.

5.2. L2: Attack Detection System. To present a viable approach for attack detection during the second stage of the system's working, the data generated from the attack simulation phase were analyzed using the "*wrapper method.*"

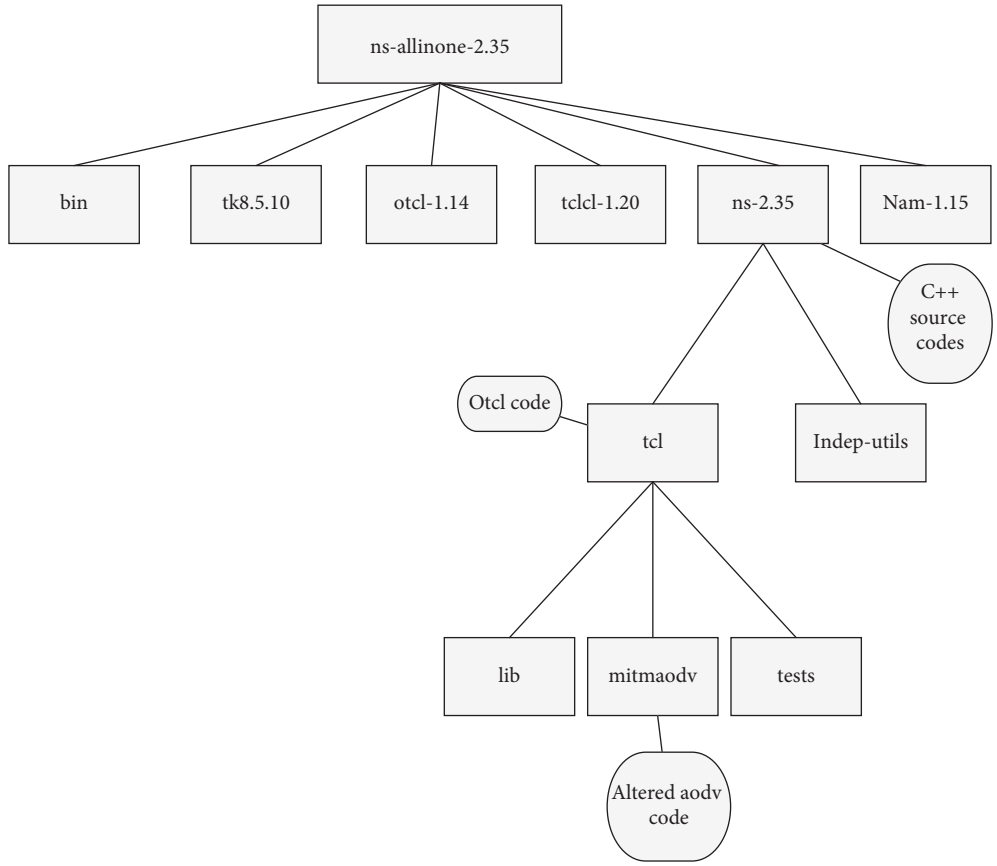

Figure 5: Revised directory structure of ns-allinone-2.35 package.

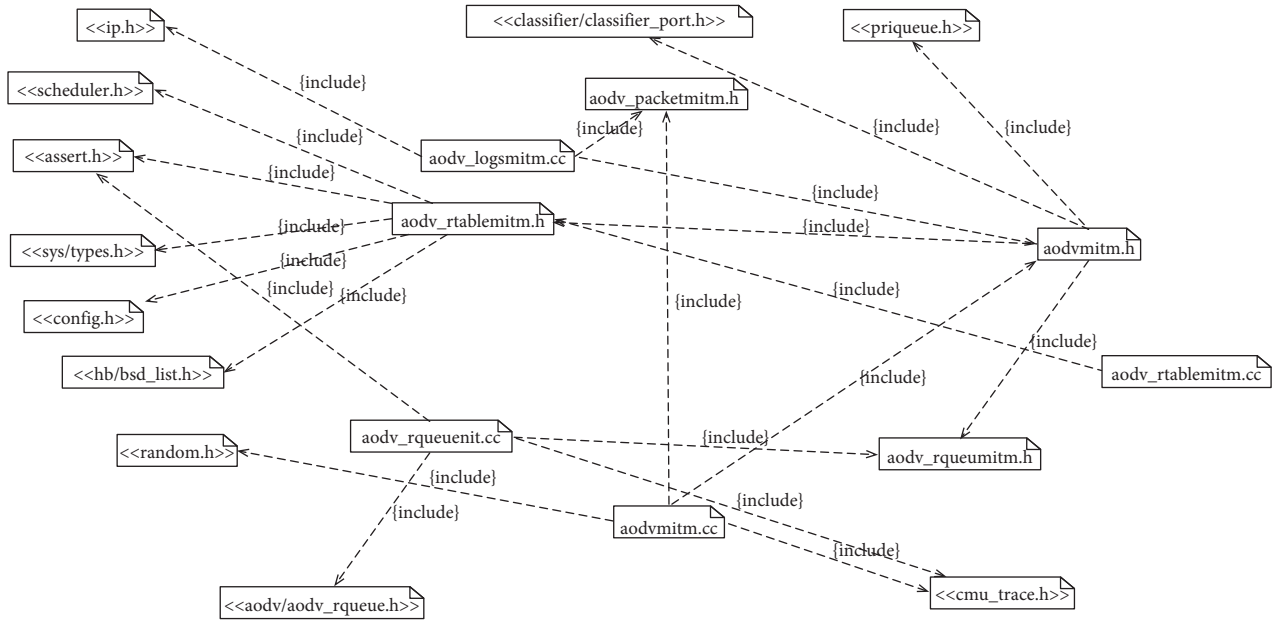

Figure 6: Component diagram for *aodvmitm* package.

The results from this analysis phase were then decomposed into the "*dataCleanerBetter.pl*" Perl script that was used in the feature extraction stage of attack detection and classification.

5.2.1. Feature Extraction and Machine Learning (Wrapper Method). There are two popular methods used in feature selection, namely, (1) the filter method (more suited for data mining) and (2) the wrapper method (more suited for

```
#PNB: added by me for protoname
AODV_MITM #for implementing man in the middle attack using the AODV protocol
{
   set allhdrs [regsub -all {#.*?\n} $protolist \n]; # strip comments from above
   foreachprot $allhdrs {
      add-packet-header $prot
}
```

Code Listing 1: Packet registration in ns-packet.tcl.

```
#PNB: a hack to satisfy the procedure node's call to man-in-the-middle attack on using modified aodv protocol
   AODV_MITM {
      set ragent [$self create-aodvmitm-agent $node]
   }
   DSDV {
      set ragent [$self create-dsdv-agent $node]
   }
   DSR {
      $self at 0.0 "$node start-dsr"
   }
```

Code Listing 2: Additions to ns-lib.tcl file.

```
#PNB: function definition for this function's call online 626
Simulator instproc create-aodvmitm-agent {node} {
   #Create Aodvmitm routing agent
   set ragent [new Agent/AODV_MITM [$node node-addr]]
   $self at 0.0 "$ragent start"
   $node set ragent_ $ragent
      return $ragent
   }
```

Code Listing 3: Additions to ns-lib.tcl files.

Table 1: Parameters used in both "*Setdest*" and "*Cbrgen*" generated files.

"Setdest" generated parameters for simulation	"Cbrgen" generated parameters for simulation
Number of nodes (n): 7 to 20	Type (type): cbr
Pause time (p): 1	Number of nodes (nn): 7 to 20
Maximum speed (M): 20	Seed (seed): 2.0
Simulation time (t): 500	Maximum connection (mc): 9
Max X (x): 750	Rate (rate): 10.0
Max Y (y): 750	

machine learning). Using the WEKA software tool, it was observed that the wrapper method was the best, because the starting number of features—twenty-six (26) in total—which were extracted by the Perl script "*dataCleanerBetter.pl*" was relatively small. Primarily, the problem at that stage was a machine learning problem as opposed to a data mining problem. So, the wrapper method helped to identify the features that could offer better classification accuracy. Conceptually, the wrapper method creates all possible subsets from the feature vector and then uses a classification algorithm to induce classifiers from each feature in each subset. It would give the set of features in which the classification algorithm (multilayer perceptron in this case) performs the best. The search technique adopted by the evaluator (*ClassifierSubsetEval* was chosen) in its quest to find the best classifier could be a depth-first search, a breadth-first search, a random search, or a hybrid search. The *BestFirst* search method was used in this case. However, before using the multilayer perceptron, features that by inspection added no new information (remained constant in the values they presented) was eliminated. They were *Pn*, *Po*, *Nz*, *Nw*, *Ne*, *Nl*, *Ma*, *Md*, *Ms*, and *If*. The pruned vector of features contained *Hs*, *Hd*, *Id*, *Ii*, *Il*, *ls*, *lt*, *lv*, *Mt*, *Ni*, *Nx*, *Ny*, *Pf*, *Pi*, *t*, and *PM*, with *It* as the classification feature.

To ensure further certainty as to which features contributed the most information, even before the selection of the classification algorithm, clustering was done using the simple K-means algorithm. To facilitate the necessary stages

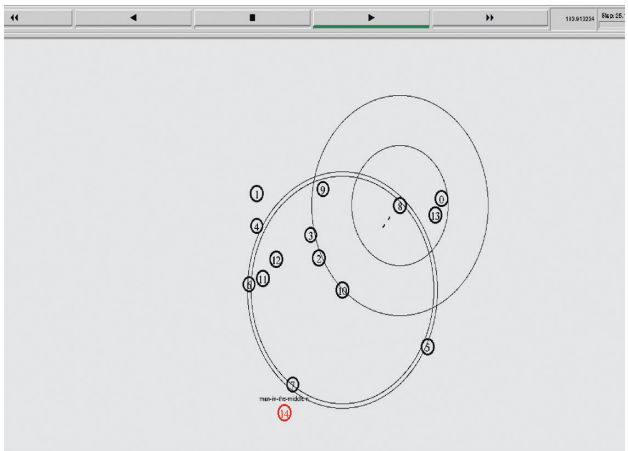

Figure 7: Simulation attack with 14 genuine nodes and 1 malicious node.

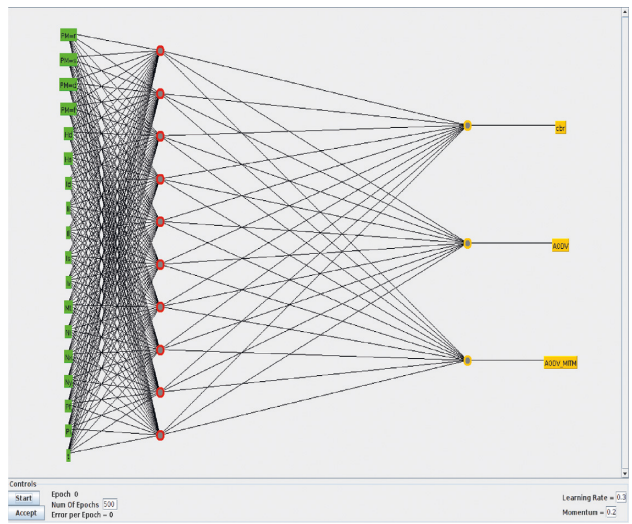

Figure 8: Initial ANN model generation for classification.

of clustering and subsequent classification, the Perl scripts were used to extract the essential features, as well as preprocess the files obtained from the feature extraction phase for possible presentation to the *MitmProtectorWithWeka* Java program —the software solution meant for attack detection.

5.3. L3: Attack Recovery System. Having identified the necessary features for possible attack detection, the information obtained was used to develop a Java software tool named "*MitmProtectorWithWeka*." That was an ANN tool that uses the identified features to check log files of network simulations, blacklists the attacking nodes based on the information from the ANN classification, and then reconfigures the network while eliminating the offending nodes. The most common structure for multilayer perceptron neural networks has three layers with full interconnections. The input layer nodes are passive relaying information from their single inputs to their multiple connections in the hidden layer. In effect, the hidden layer and the output layer are active modifying the signal flows to generate corresponding outputs. The action of this neural network is determined by the weights applied in the hidden and output nodes. The initial ANN model generated during simulation is depicted in Figure 8 with its learning rate of 0.3 and momentum of 0.2 and running different epochs. The multilayer perceptron model with the above parameters was used in the experiments using NS2 simulator with the Network Animator (NAM). The main programming languages used for the implementation were Perl, C/C++, and Java with the combination of all the different scripts and executable codes being done via shell scripts. The Linux Ubuntu 13.10 operating system served as the platform of choice for all the coding and simulations. The Java implementation of the final software solution—MitmProtectorWithWeka.jar—makes use of the WEKA API (Application Programming Interface). It automatically selects the correct features and runs the trained software against any user-configured attack detection log file. It is from this log file that it generates a possible blacklist, for automatic attack detection and system reconfiguration.

To evaluate the performance of the system, the following performance metrics for machine learning algorithms were used, namely,

$$\begin{aligned} \text{recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}}, \\ \text{precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ \text{accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \\ F - \text{measure} &= 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}, \end{aligned} \tag{1}$$

where TP = true positives: number of examples predicted positive that are actually positive; FP = false positives: number of examples predicted positive that are actually negative; TN = true negatives: number of examples predicted negative that are actually negative; and FN = false negatives: number of examples predicted negative that are actually positive.

The results presented per node for the multilayered perceptron-based classification algorithm for the classes under consideration: *cbr*, *AODV*, and *AODV_MITM* in the various instances have a very high true positive rate (TP) and a very low false positive rate (FP).

This shows that the system (software) performs well when used in time signature fingerprinting of packets from attacking nodes. It can viably distinguish between genuine and malicious sourced packets. Stress testing for multiple nodes shows appreciably high percentages per node for correctly classified instances, with relatively lower incorrectly classified instances. The mean absolute error as well as root-mean-square (RMS) errors are quite low, each increasing gradually as node number increases. That means the system performs better for lower node numbers, and its performance declines ever slightly with an increase in the number of nodes. That provided some great insight into future research.

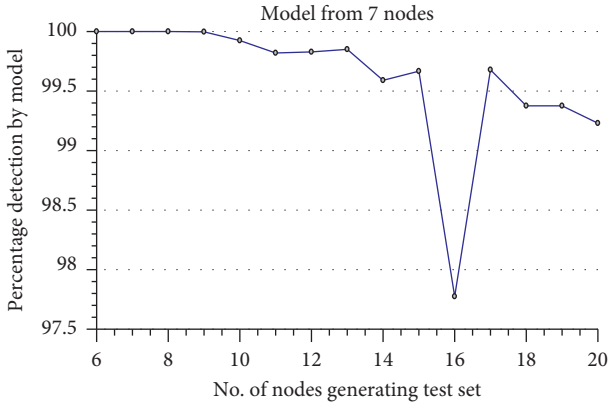

Figure 9: Simulation results for 7 nodes.

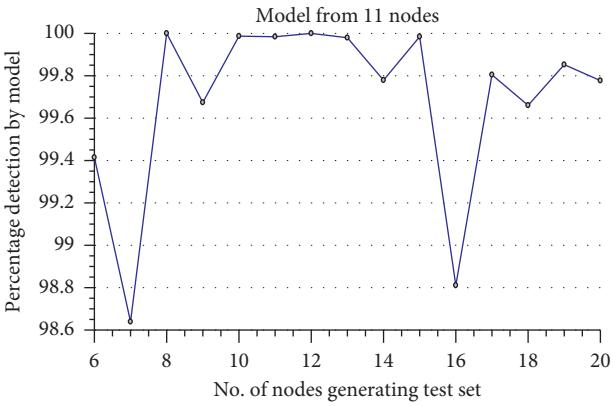

Figure 10: Simulation results for 11 nodes.

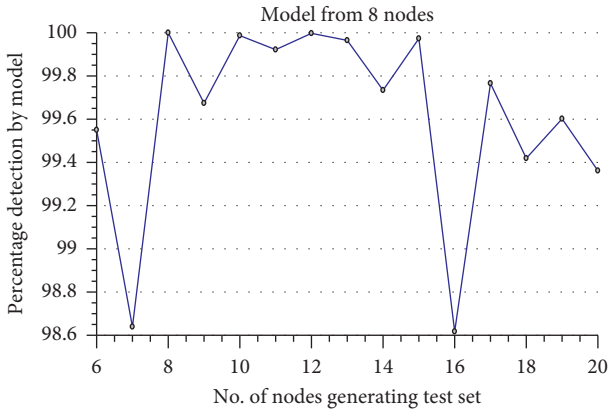

Figure 11: Simulation results for 8 nodes.

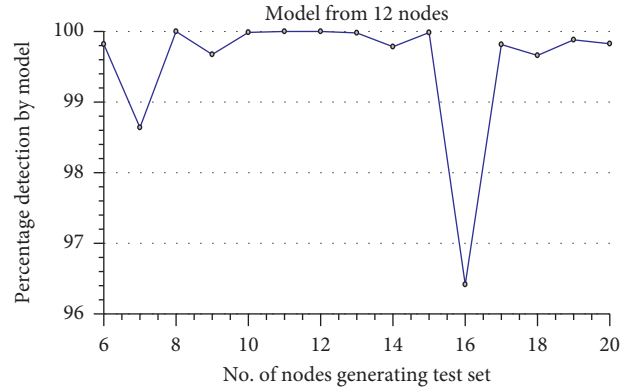

Figure 12: Simulation results of 12 nodes.

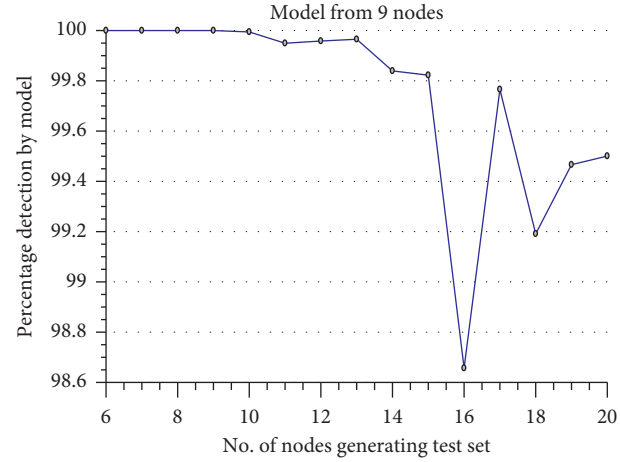

Figure 13: Simulation results for 9 nodes.

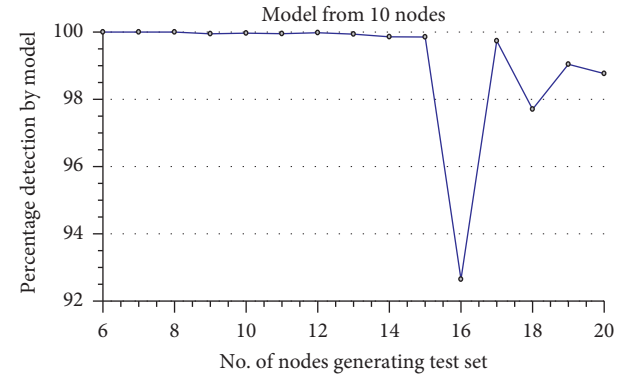

Figure 14: Simulation results for 10 nodes.

6. Results and Discussion

Figures 8–21 present the plot of correctly classified instances for each model generated. The models for detecting attacks were created with node numbers six (6) to twenty (20), and the results are presented in the figures above. Additionally, it is observed that there is a sharp drop in the percentage of correctly classified instances when the dataset presented to the model for testing originated from using sixteen (16) participatory nodes in the network. This is because the highest number of discarded/unknown instances occurred for the results with 16 nodes during the preprocessing phase. The portion of the plot revealing relatively high percentages of detection is because it had a far lesser number of unknown instances.

Therefore, it can be concluded that regardless of the number of nodes used to generate the detection model, detection is still influenced by the number of unknown

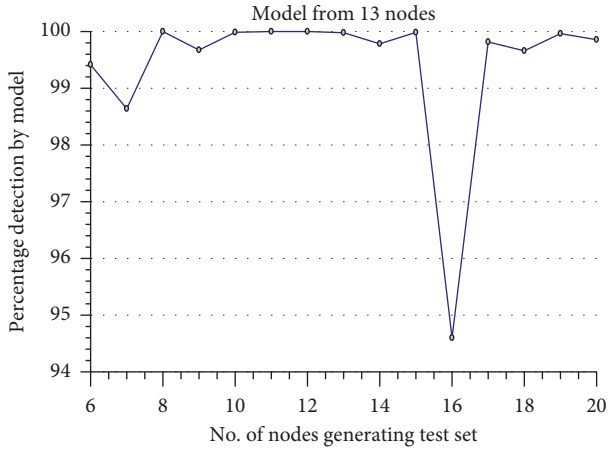

Figure 15: Simulation results for 13 nodes.

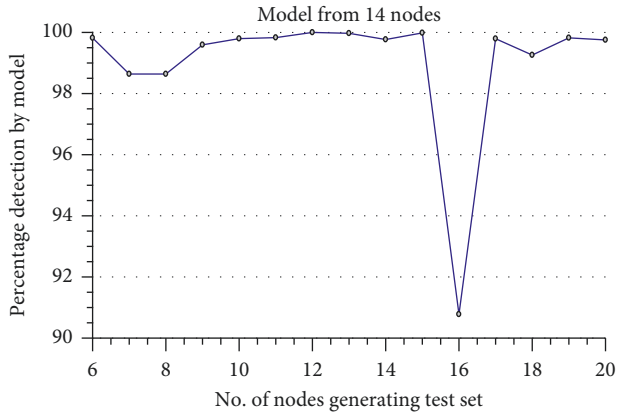

Figure 16: Simulation results for 14 nodes.

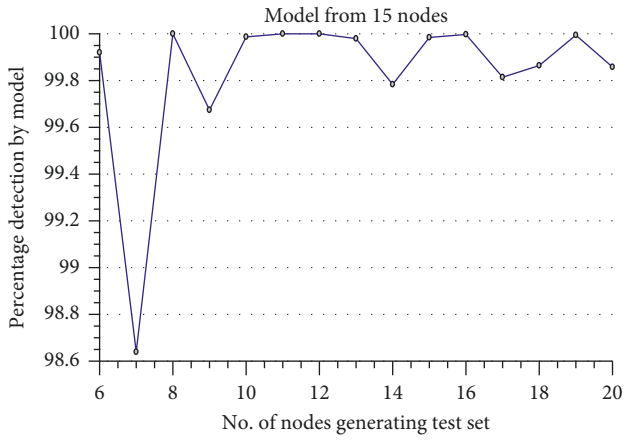

Figure 17: Simulation results for 15 nodes.

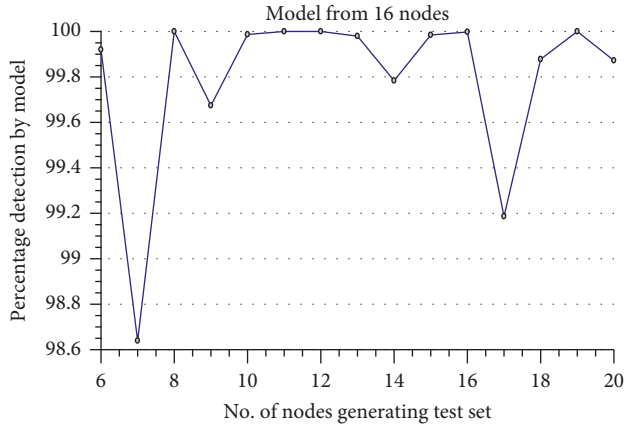

Figure 18: Simulation results for 16 nodes.

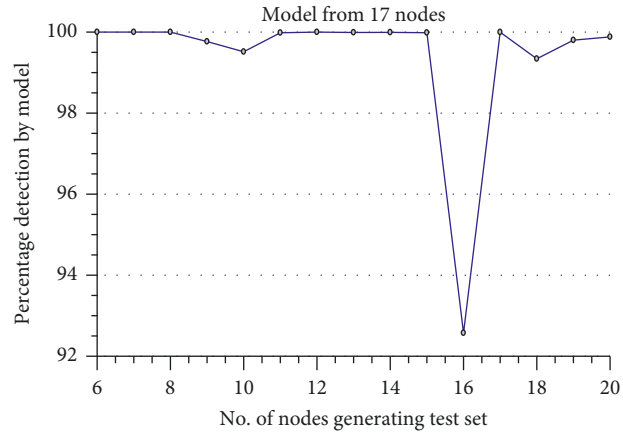

Figure 19: Simulation results for 17 nodes.

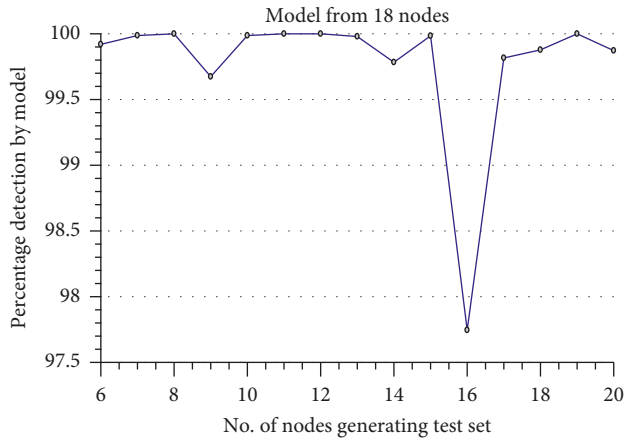

Figure 20: Simulation results for 18 nodes.

instances. A reduction in this figure results in a significant and corresponding increase in detection rates.

7. Conclusion and Future Extension

The computational times for the simulation for the different nodes are given in Table 2. From the results presented in Tables 2 and 3 and Figures 8–20, it can be observed that the model generated for the classification performs admirably well, having been tested for different node numbers–each recording very high percentages, for correctly classified instances, at reasonable root-mean-square error (RMS) values. The implication is that such an approach could be used by system administrators and security experts to time

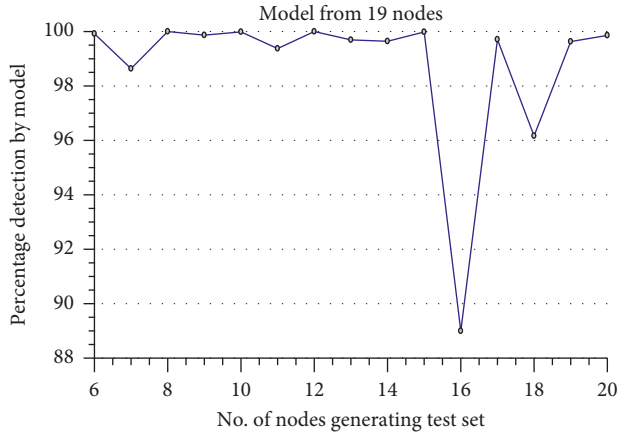

Figure 21: Simulation results for 19 nodes.

Table 2

Number of nodes	Total number of instances	Number of ignored class unknown instances	Correctly classified instances	Computational time (s)
7	29971	15609	12730 (88.6367%)	0.02
8	29971	14492	14021 (90.5808%)	0.00
9	29971	9759	17494 (86.5525%)	0.00
10	29971	14033	14029 (88.0223%)	0.02
11	29971	15826	12884 (91.0852%)	0.02
12	29971	14488	14469 (93.4509%)	0.02
13	29971	15992	11684 (83.5825%)	0.05
14	29971	15875	12606 (89.4296%)	0.05
18	29971	15407	11582 (79.5249%)	0.01

Table 3: Confusion matrices of different numbers of node configuration scenarios for stress testing with computational times.

	TP rate	FP rate	Precision	Recall	*F*-measure	ROC area	Class
	1	0	1	1	1	1	Cbr
17 nodes	0.62	0	1	0.62	0.77	0.78	AODV
	0	0.05	0	0	0	—	AODV_MITM
Weighted average	0.95	0	1	0.95	0.97	0.971	
	1	0	1	1	1	1	Cbr
18 nodes	0.62	0	1	0.615	0.76	0.781	AODV
	0	0.079	0	0	0	—	AODV_MITM
	1	0	1	1	1	0.999	Cbr
19 nodes	0.547	0	1	0.547	0.707	0.748	AODV
	0	0.148	0	0	0	—	AODV_MITM
Weighted average	0.852	0	1	0.852	0.904	0.916	
	1	0	1	1	1	1	Cbr
20 nodes	0.658	0	1	0.658	0.794	0.794	AODV
	0	0.039	0	0	0	—	AODV_MITM
Weighted average	0.961	0	1	0.961	0.977	0.98	

signature fingerprint popular software used by attackers by using these fingerprints to train the ANN and then detect attacks in the process. Thus, applying ANN in time signature fingerprinting of MITM attacks is efficient and effective for detection, classification, and control of attacks. Future research could involve more of the unknown instances—which had to be discarded from the model recorded in attack decision making. One of the difficulties that were noticed during the training phase of the ANN algorithm was the number of instances that were not used because they were not distinguishable. The results provided based on simulation ignored the instances that do not contribute to the overall detection of attacks. Table 2 results give an insight into the computational times realized for some models used in the detection and reconfiguration of the network after attack detection.

To improve the detection rate of any future model should be focused on getting higher true positive rates out of the

confusion matrix for AODV_MITM packets, since this would suggest better detection rate for attacks. One of the possible ways could be to increase the time lapse for packet time transmission between promiscuous nodes which send out AODV_MITM packets.

The original work could also be extended to include popular commercial or open-source spyware for performing MITM attacks. Time signature fingerprinting them and adding their learned time signature fingerprints could be vital for security researchers in the detection and prevention of future attacks. Research plans are in place to compare the results to other well-established systems in the MANET security ecosystem for attack detection and prevention since at the time of the writing of this paper, there was no known similar standardized testing method offered by a simulated environment, suggesting a possible way for results replication.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

Portions of this paper were presented at the GIC 2017 meeting. The work was based on the thesis by the authors.

Conflicts of Interest

The received funding did not lead to any conflicts of interest regarding the publication of this manuscript.

Acknowledgments

The authors of this paper wish to acknowledge the Carnegie Corporation of New York through the University of Ghana, under the UG-Carnegie Next Generation of Academics in Africa project for financially supporting this research work. The help has been immeasurable in the realization of this research.

References

[1] P. Goyal, V. Parmar, and R. Rishi, "MANET: vulnerabilities, challenges, attacks, application," *International Journal of Computational Engineering and Management*, vol. 2011, no. 11, pp. 32–37, 2011.

[2] J. Pearlman and P. Rheingans, "Visualizing network securityevents using compound glyphs from a serviceriented perspective," J. R. Goodall, G. Conti, and K. Ma, Eds., in *Proceedings of the Workshop on Visualization for Computer Security (VizSEC 2007)*, pp. 131–146, Springer Berlin Heidelberg, Berlin, Germany, 2008.

[3] D. Hurley-Smith, J. Wetherall, and A. Adekunle, "SUPERMAN: security using pre-existing routing for mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 10, pp. 2927–2940, 2017.

[4] P. Hari, V. K. Shukla, and P. R. Verma, "An innovative approach for security on Mobile Ad-Hoc Network," in *Proceedings of 2015 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, pp. 759–765, Kumarakoil, India, December 2015.

[5] J.-M. Chang, P.-C. Tsou, I. Woungang, H.-C. Chao, and C.-F. Lai, "Defending against collaborative attacks by malicious nodes in MANETs: a cooperative bait detection approach," *IEEE Systems Journal*, vol. 9, no. 1, pp. 65–75, 2015.

[6] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Wormhole attacks in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 370–380, 2006.

[7] J. Ben Othman and M. Ayaida, "Special issue on last advances on QoS and security in wireless networks," *Journal of Communications and Networks*, vol. 16, no. 4, pp. 358–362, 2014.

[8] RSA, "Making sense of man-in-the-browser attacks: threat analysis and mitigation for financial institutions," in *RSA White Paper*, RSA LLC, Bedford, MA, USA, 2010.

[9] The Department of Defense Cyber Strategy, *The DOD Cyber Strategy*, US Department of Defense, Arlington, VA, USA, 2015, http://www.defense.gov/Portals/1/features/2015/0415_cyber-strategy/Final_2015_DoD_CYBER_STRATEGY_for_web.pdf.

[10] J. Schmidhuber, "Deep learning in neural networks: an overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[11] Z. Moradi and M. Teshnehlab, "Intrusion detection model in MANETs using ANNs and ANFIS," in *Proceedings of CSIT 2011 International Conference on Telecommunication Technology and Applications*, Vol. 5, IACSIT Press, Singapore, 2011.

[12] K. El Defrawy and G. Tsudik, "ALARM: anonymous location-aided routing in suspicious MANETs," *IEEE Transactions on Mobile Computing*, vol. 10, no. 9, pp. 1345–1358, 2011.

[13] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[14] H. Yang, H. Luo, F. Ye, S. Lu, and L. Zhang, "Security in mobile ad hoc networks: challenges and solutions," *IEEE Wireless Communications*, vol. 11, no. 1, pp. 38–47, 2004.

[15] V. Rajamanickam and D. Veerappan, "Inter cluster communication and rekeying technique for multicast security in mobile ad hoc networks," *IET Information Security*, vol. 8, no. 4, pp. 234–239, 2014.

[16] Resource Centre, "Catalogue of B.Tech. Project Reports Batch–2005-09 Abstracts," Resource Centre, Gandhinagar, India, 2010.

[17] S. Kurosawa, H. Nakayama, and N. Kato, "Detecting blackhole attack on AODV-based mobile ad hoc networks by dynamic learning method," *International Journal of Network Security*, vol. 5, no. 3, pp. 338–346, 2007.

[18] R. Thomas, K. Böhm, A. S. Ward, and E. Buchmann, "Malicious behaviour in content-addressable peer-to-peer networks," in *Proceedings of 3rd Annual Communication Networks and Services Research Conference (CNSR'05)*, pp. 319–326, Halifax, Canada, May 2005.

[19] S. Maag, C. Grepet, and A. Cavalli, "A formal validation methodology for MANET routing protocols based on nodes' self similarity," *Computer Communications*, vol. 31, no. 4, pp. 827–841, 2008.

[20] A. K. Rai, R. R. Tewari, and S. K. Upadhyay, "Different types of attacks on integrated MANET-Internet communication," *International Journal of Computer Science and Security*, vol. 4, no. 3, pp. 265–274, 2010.

[21] A. M. Abdalla, A. H. Almazeed, I. A. Saroit, and A. Kotb, "Detection and isolation of packet dropping attacker in

MANETs," *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 4, 2013.

[22] Y. Chen, S. Das, P. Dhar, and A. El-Saddik, "Detecting and preventing IP-spoofed distributed DoS attacks," *International Journal of Network Security*, vol. 7, no. 1, pp. 69–80, 2008.

[23] A. Mitrokotsa and C. Dimitrakakis, "Intrusion detection in MANET using classification algorithms: the effects of cost and model selection," *Ad Hoc Networks*, vol. 11, no. 1, pp. 226–237, 2013.

[24] S. Dokurer, *Simulation of Black Hole Attack in Wireless Ad-Hoc Networks*, Atilim University, Ankara, Turkey, 2006.

[25] A. Anand, H. Aggarwal, and R. Rani, "Partially distributed dynamic model for secure and reliable routing in mobile ad hoc networks," *Journal of Communications and Networks*, vol. 18, no. 6, pp. 938–947, 2016.

[26] S. Imran, R. V. Karthick, and P. Visu, "DD-SARP: dynamic data secure anonymous routing protocol for MANETs in attacking environments," in *Proceedings of 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, pp. 39–46, Avadi, Chennai, India, May 2015.

[27] http://ugspace.ug.edu.gh/handle/123456789/7319.

[28] J. Chung and M. Claypool, *NS by Example*, Worcester Polytechnic Institute, Worcester, MA, USA, 2002.

[29] G. Holmes, A. Donkin, and I. H. Witten, "WEKA: a machine learning workbench," in *Proceedings of Australian New Zealand Intelligent Information Systems Conference (ANZIIS'94)*, pp. 357–361, Brisbane, QLD, Australia, 1994.