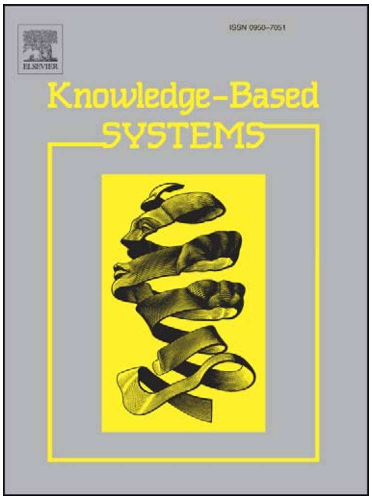

Evaluation of Classification Algorithms for Intrusion Detection in MANETs

Sergio Pastrana, Aikaterina Mitrokotsa, Agustin Orfila, Pedro Peris-Lopez

PII: S0950-7051(12)00187-6
DOI: http://dx.doi.org/10.1016/j.knosys.2012.06.016
Reference: KNOSYS 2338

To appear in: *Knowledge-Based Systems*

Received Date: 28 February 2012
Revised Date: 26 May 2012
Accepted Date: 28 June 2012

Please cite this article as: S. Pastrana, A. Mitrokotsa, A. Orfila, P. Peris-Lopez, Evaluation of Classification Algorithms for Intrusion Detection in MANETs, *Knowledge-Based Systems* (2012), doi: http://dx.doi.org/10.1016/j.knosys.2012.06.016

Evaluation of Classification Algorithms for Intrusion Detection in MANETs

Sergio Pastrana[a,1], Aikaterina Mitrokotsa[b], Agustin Orfila[a], Pedro Peris-Lopez[a]

[a] *Computer Science Department, Carlos III University of Madrid, Leganes, Spain*
[b] *Security and Cryptography Laboratory (LASEC), School of Computer and Communication Sciences, EPFL, Switzerland*

Abstract

Mobile Ad-hoc Networks (MANETs) are wireless networks without fixed infrastructure based on the cooperation of independent mobile nodes. The proliferation of these networks and their use in critical scenarios (like battlefield communications or vehicular networks) require new security mechanisms and policies to guarantee the integrity, confidentiality and availability of the data transmitted. Intrusion Detection Systems used in wired networks are inappropriate in this kind of networks since different vulnerabilities may appear due to resource constraints of the participating nodes and the nature of the communication. This article presents a comparison of the effectiveness of six different classifiers to detect malicious activities in MANETs. Results show that Genetic Programming and Support Vector Machines may help considerably in detecting malicious activities in MANETs.

Key words: MANET, Intrusion Detection, Genetic Programming, Classification Algorithms

1. Introduction

Due to the inherent vulnerabilities of wireless networks, new security measures need to be developed to efficiently safeguard them. This work fo-

Email addresses: spastran@inf.uc3m.es (Sergio Pastrana), katerina.mitrokotsa@epfl.ch (Aikaterina Mitrokotsa), adiaz@inf.uc3m.es (Agustin Orfila), pperis@inf.uc3m.es (Pedro Peris-Lopez)

[1] Corresponding author: spastran@inf.uc3m.es, +34 91 624 6260

cuses on the detection of malicious activities in MANETs. Proposed ideas for intrusion detection in these networks are required to achieve a trade-off between accurate detection and limited consumption of resources [1] or the lack of central management and mobility of nodes [2]. This article presents a comparison of different classification algorithms applied to detect intrusions in MANETs. These algorithms can help to discriminate “normal” against “intrusive” behaviour effectively. We use six well-known classification algorithms, using labelled datasets obtained from a simulated environment. The comparison is fairly performed as several hyper-parameters were tuned and the experiments have been performed with datasets generated under various traffic conditions regarding the network mobility and the number of malicious nodes. We focus on detecting four different types of attacks: Black Hole [3], Forging [4], Packet Dropping [5] and Flooding [6].

In Section 2 we present the motivation of our work. Section 3 introduces a review of the state of the art on intrusion detection in MANETs. Section 4 explains the details of the experimental setup and Section 5 presents the results obtained. Finally, we conclude in Section 6.

1.1. Contribution

This article extends a previous work [7] that compares how effective intrusions in MANETs are detected by different classification algorithms, namely, Naïve Bayes, Gaussian Mixture Model (GMM), Multilayer Perceptron (MLP), Linear Model and Support Vector Machines (SVM). Accordingly, two main contributions are presented. First, we analyse the promising behaviour of a new method based on Genetic Programming. Second, different probabilities of attacks are considered tackling a wide set of scenarios.

2. Motivation

2.1. Wireless technology

Wireless networks use the open air medium as communication channel and electromagnetic waves to send information between participants. Nodes in wireless networks can communicate with every other node located within a specific distance, called transmission range. When a node wants to send a packet to another node that does not belong in its one-hop neighbourhood then it has to rely to intermediate nodes to forward the packets to the final destination. Thus, efficient routing protocols are required in order to optimize the communication paths. Security issues in wireless communication may also have a serious impact in other types of network architectures since several network architectures use wireless channels. For instance, an architecture using the 802.11 standard (usually referred as WIFI or WLAN networks) uses a fixed infrastructure that communicates with other networks using a wired communication, but communicates with the nodes of its own network using a wireless channel. This architecture requires all the nodes to be placed within the transmission range of the fixed infrastructure (access point), and any problem regarding this central point may affect the entire network.

Wireless ad-hoc networks or Mobile Ad-hoc Networks (MANETs) do not use a fixed infrastructure and all the nodes belonging to the network may be mobile. There is no central node acting as an access point, and mobile nodes share the responsibility of the proper functionality of the network, since a collaborative behaviour is required.

2.2. Security in MANETs

The intrinsic nature of MANETs provokes the emergence of new security risks, while some existing vulnerabilities in wired networks are accentuated. The use of security technologies developed for wired networks in order to safeguard wireless networks is neither direct nor easy to perform. In the absence of a wire connecting the nodes, any malicious node may access the network without physical restrictions. In order to prevent fraudulent outsiders entering the network, cryptographic algorithms can be used to authenticate the nodes. However, more complicated problems arise when an internal benign node is compromised, that is, if any attacker impersonates the identity of a node that is authorized in the network. Since the functionality of the network is typically based on a complete confidence between the participants, a malicious node impersonating a trusted node may cause a serious security bridge. Most of the attacks in mobile environments focus on routing protocols. These protocols were firstly designed to be efficient without taking into account the security issues. They usually need the cooperation between the participants and assume confidence between them. Nevertheless, a malicious node may modify its supposed benign functionality disturbing the overall behaviour of the protocol. Below, we present a list of attacks that we have considered in order to evaluate the effectiveness of the employed algorithms for the problem of intrusion detection.

- Packet Dropping attack: In this attack, the attacker rejects Route Error packets leading legitimate nodes to forward packets in broken links [5].

- Flooding attack: The malicious node broadcasts forged Route Request packets randomly to all nodes every 100 msec in order to overload the network [6].

- Black Hole attack [3]: In this attack a malicious node advertises itself as having the shortest path to other nodes of the network. Nevertheless, as soon as it receives packets destined for other nodes, it drops them instead of forwarding to the final destination. In our simulation scenario, each time a malicious black-hole node receives a Route Request packet it sends a Route Reply packet to the destination without checking if it really has a path towards the selected destination. Thus, the black-hole node is always the first node that responds to a Route Request packet. Moreover, the malicious node drops all Route Reply and Data packets it receives if the packets are destined to other nodes.

- Forging attack [4]: A malicious node modifies and broadcasts to the victim node Route Error packets leading to repeated link features.

3. State of the art

3.1. Intrusion detection in MANETs

Intrusion Detection Systems (IDS) are software or hardware tools (even a combination of both) that automatically scan and monitor events in a computer or network, looking for intrusive evidence [8]. When designing an IDS to be used in a MANET, some considerations must be taken into account. There are several differences in the way the detection engine must behave with respect to a wired network IDS. A rather complete survey about this topic can be found in [2], where Anjum *et al.* present the main challenges to secure wireless mobile networks. More recently, Sen *et al.* [9] have presented a survey about existing intrusion detection approaches for MANETs.

Traditional anomaly-based IDSs use predefined "normality" models to detect anomalies in the network. This is an approach that cannot be easily deployed in MANETs, since the mobility and flexibility of MANET nodes, make hard the definition of "normal" and "malicious" behaviour. Furthermore, the mobility of nodes leads to changes of the network topology, increasing the complexity of the detection process. Additionally, since the MANET nodes have no fixed location, there is no central management and/or monitoring point where an IDS could be placed. This implies that the detection process may be distributed into several nodes, as well as the collection and analysis of data. Consequently IDS are classified into collaborative or independent (non-collaborative) [9]. Independent IDSs are composed of IDS agents placed into the nodes of the network and being responsible for monitoring their neighbours and sending alarms whenever they detect any suspicious activity. The major problem of this architecture is deciding the location of the IDS agents, since nodes are mobile, and some zones of the network may not be monitored (for example, if the node hosting an IDS agent of one zone moves to another, the first remains uncovered). Another problem is that some resources such as bandwidth, CPU and/or power are scarce in these environments. Therefore, nodes hosting the IDS agents should be those having more resources and moreover, a larger transmission range. Maximising the detection rate subject to resource limitation is an NP-complete problem and some algorithms have been proposed to approximate the solution [2].

Several IDS architectures have been proposed to be used in mobile networks. Zhang *et al.*, initially in 2000 [10], and later in 2003 [11], proposed a distributed and collaborative detection architecture. Every node in the network monitors their local neighbours, locally and independently, to detect any sign of intrusion. The key idea is that they may share information to perform this intrusion search. Each IDS agent is structured in several pieces or modules. Initially a data collection module gathers audit traces and activity logs. Then, a local detection engine analyses the data to look for local anomalies. Two modules are responsible for performing the response actions: the local and global response modules. To share information, an extra secure communication module is used to provide trusted communications. In these approaches [10, 11], Zhang *et al.* use classifiers to detect anomalies. They use entropy and conditional entropy to describe the characteristics of "normal" traffic and classification algorithms to build models of "normal" behaviour. Therefore classifiers are trained using "normal" data, to predict what is normally the next event given the previous n events. If a detector node monitors an event which is not what the classifier has predicted, an alarm is triggered.

Huang *et al.* [12] presented a cluster-based IDS, in order to combat the resource constraints of MANETs. The authors use a set of statistical features obtained from routing tables and apply a classification decision tree algorithm, the C4.5, in order to discriminate "anomalous" against "normal" traffic. This approach allows the identification of the source of the attack, if the attack occurs within one-hop. Later, in 2004 [13] they proposed a hybrid system where they use both specification-based and anomaly-based detection, by using a taxonomy of anomalous activities and a finite state machine, which represents the correct behaviour of the Ad-hoc On Demand Distance Vector (AODV) [14] protocol.

In 2003 Karchirski and Guha [15] proposed the use of multiple collaborative sensors, where each sensor acts as a lightweight mobile agent. Each agent has a different role: network monitoring, host monitoring, decision-making and action-taking. The nodes are divided into clusters, and each cluster has a head node which monitors packets. Nodes vote to select their cluster head, based on the connectivity data received after a broadcast step. Karchirski and Guha focus on minimizing the use of resources by the nodes in the network. However, they do not give details about how the detection process is performed.

Sun *et al.* [16] have also dealt with the problem of cooperativeness between nodes and presented a non-overlapping zone-based IDS. In their approach, the nodes of the network are grouped into zones, such that some of the internal nodes of a zone act as gateways to other zones. The nodes of the network use Markov Chains to detect intrusions and they send alarms to their corresponding gateway when they detect some abnormal activity, using the proposed MANET Intrusion Detection Message Exchange Format (MIDMEF).

Recently, Su [17] has proposed a cooperative intrusion detection system where some nodes monitor their neighbours in order to detect packets that are suspicious of being part of a Black Hole attack. When the number of such suspicious packets sent by a node exceeds a threshold, the detector node broadcasts a block message to all the nodes in the network. This block message is firstly authenticated with the ID of the detector node, and carries information indicating that the packets sent by the malicious node should be ignored. Our approach is different since we do not detect Black Hole attacks with a specific threshold, but we study the use of classifiers.

Sen *et al.* [18] has presented different evolutive approaches to detect intrusions. More precisely, the authors use simulated networks to obtain the data they use to evolve the programs, implementing different attacks. First, in [18] a grammatical approach is used to detect Packet Dropping , Flooding and Route Disruption attacks. They achieve good detection rates for the three types of attacks, but with a rather high false positive rate in the Packet Dropping and Flooding attack. They argue that this is due to packet losses that usually occurs in these networks, and differentiating packet losses from malicious droppings is not an easy task. Secondly, in [19] they use Genetic Programming with a multi-objective approach to obtain programs that maximize the detection rate and minimise both the false positive rate and the energy consumption, which is one of the main constraints in MANETs. They evaluate their approach for two types of attacks, the Flooding attack and the Route Disruption attack. In both works, different intrusion detection approaches are employed for each kind of attack and again almost all the attacks are detected. Our work is different in several ways. First, although the attacks are similar, the behaviour of malicious nodes in our experiments is different. For instance, in their Packet Dropping attack, malicious nodes drop Data packets whereas our malicious nodes drop Route Error packets, that is a more general situation that reflects how cheating nodes maintain broken links. Second, Sen et al. work evolve one classifier for each attack (resulting in 3 different programs), while in our work we only generate one classifier to detect all the attacks studied, which entails a lesser consume of resources and better performance for constrained devices. Third, some experimental settings used in our work are different. For example, we use the GlomoSim simulator program whereas they used NS-2, the pause time between movements of nodes is different (5, 20 and 40 vs 0, 20, 400 and 700), etc.

3.2. Classification models

The classification models we have considered are six well known classifiers i.e., the MultiLayer Perceptron (MLP), the Linear classifier, the Gaussian Mixture Model (GMM), the Naïve Bayes classifier, Support Vector Machine (SVM) model and Genetic Programming (GP) algorithms employed as classifiers.

An instance of an MLP can be considered as a function $g : \mathcal{X} \rightarrow \mathcal{Y}$, where g can be defined as a composition of other functions $z_i : \mathcal{X} \rightarrow \mathcal{Z}$. This decomposition can be written as $g(x) = Kw'z(x)$ where $x \in \mathcal{X}$, w is a parameter vector and K denotes a kernel and the function $z(x) = [z_1(x), z_2(x), ...]$ is called *hidden layer*. For each of those hidden layers, it holds $z_i(x) = K_i(v_i'x)$ where each v_i is a parameter vector, $V = [v_1, v_2, ...]$ denotes the parameter matrix of the hidden layer and K_i denotes an arbitrary kernel. For the problem of intrusion detection we use an MLP m, as a model for the conditional probability given the observations i.e., :

$$\mathbb{P}(\mathcal{Y} = y|\mathcal{X} = x, \mathcal{M} = m), \quad y = g(x).$$

If there is no hidden layer then it holds $z_i = x_i$ and the model m corresponds to the Linear model.

The GMM model is used to model the conditional observation density of each class y i.e., :

$$\mathbb{P}(\mathcal{X} = x|\mathcal{Y} = y, \mathcal{M} = m).$$

This is achieved by using a separate set of mixtures U_y for modeling the observation density of each class y. Thus, for a given class y the density at each point x is calculated by marginalizing over the mixture components $u \in \mathcal{U}_y$, for the class, i.e., :

$$\mathbb{P}(\mathcal{X} = x|\mathcal{Y} = y) = \sum_u \mathbb{P}(\mathcal{X} = x|\mathcal{U} = u)\, \mathbb{P}(\mathcal{U} = u|\mathcal{Y} = y).$$

The likelihood function $\mathbb{P}(\mathcal{X} = x|\mathcal{U} = u)$ has a Gaussian form with parameters the covariance matrix $\sum_u$ and the mean vector μ_u, while the term $\mathbb{P}(\mathcal{U} = u|\mathcal{Y} = y)$ represents the component weight. Finally, by estimating $\mathbb{P}(\mathcal{Y} = y)$ from the data, we obtain the conditional probability given the observations, i.e., :

$$\mathbb{P}(\mathcal{Y} = y|\mathcal{X} = y) = \frac{1}{\mathcal{Z}}\, \mathbb{P}(\mathcal{X} = x|\mathcal{Y} = y)\, \mathbb{P}(\mathcal{Y} = y)$$

where $\mathcal{Z} = \sum_y \in \mathcal{Y}\, \mathbb{P}(\mathcal{X} = x|\mathcal{Y} = y)\, \mathbb{P}(\mathcal{Y} = y)$ does not depend on y.

The Naïve Bayes model can be derived from the GMM model when there is only one Gaussian mixture.

We have also used the Support Vector Machine (SVM) [20] model that uses Lagrangian methods to minimise a regularised function of the empirical classification error. The SVM algorithm finds a linear hyperplane separation with a maximal margin in this hyperspace. The points that are lying on the margin are called support vectors. The main parameters of the algorithm is c which represents the trade-off between the size of the margin and the number of violated constraints and the kernel $K(x_i, x_j)$. More precisely, we use SVMs with a gaussian kernel of the form $K(x_i, x_j) = \frac{1}{\sqrt{2\pi\sigma}} \exp(\frac{-\|x_i - x_j\|^2}{\sigma^2})$, $\forall x_i, x_j \in \mathcal{X}$, where $\mathcal{X}$ is the observation space.

Given a problem, Genetic Programming (GP) performs a heuristic search for an optimal solution over a big exploration space [21]. It manages a population of individuals (programs), randomly initialized, which are evolved regarding natural selection procedures. Each program (individual) has a tree structure where the root and intermediate nodes are functions and leafs are terminals. In each step or generation of the algorithm, programs are evaluated using a fitness function that tests the individuals, thus establishing an order of the individuals. At each generation a new population is obtained by selecting the best individuals from the previous generation (the first generation is randomly established). Some individuals are mutated (changing an internal subtree by any other) or crossed (interchanging subtrees from two different individuals). After a given number of generations the algorithm stops, and the best individual of the last generation is given as the optimal solution.

Genetic Programming has been proven to be a good paradigm in the scenario of Network Intrusion Detection Systems (NIDS) development [22, 23, 24]. The main reason is that the functions used by GP can be defined ad hoc for a particular scenario and then the algorithm selects and combines them in order to optimize the solution to the given problem. Accordingly, it is appropriate for the complex intrusion detection domain. For instance, in a recent work Kavitha et al. [25] use GP along with Neutrosophic Logic (a generalization of fuzzy logics) to generate intrusion detection rules. GP has also been used to model the internal behaviour of a NIDS considered as a black box [26].

4. Experimental setup

The main goal of this work is to analyse the performance of six classifiers when trying to detect fraudulent actions that may occur in a MANET. These classification models can be used in an independent detection engine, where nodes hosting the IDS engine (agent), work independently to detect malicious activities.

4.1. Dataset

We have simulated a mobile ad-hoc network (MANET) using the Glomosim [27] library. We assume that the network has no pre-existing infrastructure and that the employed ad-hoc routing protocol is the Ad-hoc On Demand Distance Vector (AODV) [14]. We have simulated a network of 50 nodes placed randomly within a 850×850 m^2 area. These conditions are similar to those used by Sen and Clark [18]. Each node has a radio propagation range of 250 meters and the channel capacity is equal to 2 Mbps. The nodes move according to the 'random way point model'. The minimum and maximum speed is set to 0 and 20 m/s, respectively and the pause times at

Table 1: Features of the dataset

Name	Description
RREQ Sent/Received	Number of Route Request packets sent/received.
RREP Sent/Received	Number of Route Reply packets sent/received.
RError Sent/Received	Number of Route Error packets sent/received.
Data Sent/Received	Number of bytes sent/received.
# of neighbours	Number of one-hop neighbours of each node.
PCR	Percentage of the changed routed entries in the routing table of each node.
PCH	Percentage of the changes of the sum of hops of all routing entries for each node [28].

0, 200, 400 and 700 sec. The simulation time of the experiments was 700 sec, thus a pause time equal to 0 sec corresponds to the continuous motion of the node and a pause time of 700 sec corresponds to the time that the node is stationary. Each node generates Constant Bit Rate (CBR) network traffic while the size of the packets varies from 128 to 1024 bytes what simulates a demanding and realistic case scenario. Additionally, we have generated different datasets depending on the number of malicious nodes that exist in the network 5, 15 or 25 malicious nodes while the sampling interval (i.e., time period after which data are collected from each node of the network) is 15 sec.

Finally, in order to discriminate “normal” and “attack” network activity we have used the features described in Table 1.

In intrusion detection, the prevalence of attacks, defined as the number of traces belonging to one class divided by the total number of traces, is critical to compare the effectiveness of different proposals. The original generated dataset had 80% of attacks, which means that 80 out of 100 events in the network are hostile. Although the possibilities of being attacked in mobile networks are considerably greater than in wired networks [18], this is a pessimistic view, as most of the events are malicious. In order to study our approach in different environments, we have modified the original dataset by reducing the prevalence. We have run the experiments three times to study different scenarios with different prevalence of attacks (both in training and testing subsets): 80% (the original dataset), 4% and 1%. Although these environments are simulated, they represent real situations where different

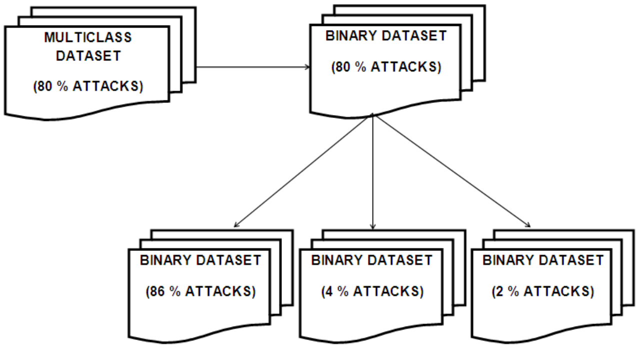

Figure 1: Setup of the datasets

security measures may be applied. On the one hand, a MANET can be physically accessible by everyone. In this case, several attackers can access the network and perform a huge number of attacks thus raising the prevalence of the attacks. On the other hand, if the access to the network is restricted with some authentication method or access control, fewer attackers may access it. This could be the case of a private MANET inside a corporation. In such a case, fewer number of attacks would be expected with respect to the total number of events in the network.

4.2. Two-class classification

Figure 1 shows a schematic view of the different datasets used. The original dataset contains labelled packets; labels may be normal (non-malicious behaviour) or any of the four attacks performed in the simulated network. In order to study two-class classification (normal or attack), we have modified the dataset by representing any attack with the label 1 (malicious trace) and the remainder, non malicious, with the label 0. With this dataset our aim is to identify hostile actions in the network without identifying the specific type of attacks. Summarizing, we have studied four different scenarios, one in which we have to identify the exact type of attack (multi-class classification) when the percentage of attacks in the dataset is 80%, and three scenarios where the classifiers may detect attacks without identifying the exact type of them (i.e. binary classification) when the percentage of attacks in the dataset is 80%, 4% and 1%.

4.2.1. Performance Comparison Metrics

When comparing algorithms, it is important to use the same measure of quality. We make the comparison of the different classifiers in terms of two different measures: the *classification error* (CE) and the *intrusion detection capability* (Cid).

For a given classification algorithm $f : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ is the observation space and $\mathcal{Y}$ is the set of classes, the *classification error* (CE) measured over an independent set D is given by:

$$\hat{E}(\mathsf{CE}|D) = \frac{1}{|D|} \sum_{d \in D} \mathsf{CE}(f(x_d), y_d)$$

where x_d is the observation of example d and y_d is its class and $\mathsf{CE}(y', y) = 0$ when $y = y'$ and 1 otherwise.

The *intrusion detection capability* (Cid) is a novel measure used in the intrusion detection domain [29]. It measures the amount of uncertainty of the input resolved once the IDS output is obtained, and takes into account the attack prevalence in the dataset besides the *Detection rate* (DR) and the *False Alarm* (FA) rate [30].

This measure takes also into consideration the effect of an imbalanced distribution. It is defined as:

$$\begin{aligned}\mathsf{Cid} = & -B \cdot \mathsf{DR} \log \frac{B \cdot \mathsf{DR}}{B \cdot \mathsf{DR} + \mathsf{DR} \cdot \mathsf{FA}} \\ & - B(1 - \mathsf{DR}) \log \frac{B(1 - \mathsf{DR})}{B(1 - \mathsf{DR}) + (1 - B)(1 - \mathsf{FA})} \\ & - (1 - B)(1 - \mathsf{FA}) \log \frac{(1 - B)(1 - \mathsf{FA})}{(1 - B)(1 - \mathsf{FA}) + B(1 - \mathsf{FA})} \\ & - (1 - B)\mathsf{FA} \log \frac{(1 - B)\mathsf{FA}}{(1 - B)\mathsf{FA} + B\mathsf{DR}}\end{aligned}$$

where B is the prevalence of attacks, DR the *detection rate* and FA the *false alarm rate*, defined as:

$$\mathsf{DR} = \frac{\mathsf{TP}}{\mathsf{TP} + \mathsf{FN}}, \qquad \mathsf{FA} = \frac{\mathsf{FP}}{\mathsf{TN} + \mathsf{FP}} \tag{1}$$

where TP, TN, FP, FN, denote the number of true (TP, TN) and false (FP, FN) positives and negatives respectively.

4.3. Multiclass classification

As previously stated, a GP execution produces programs that are intended to solve a specific problem. As the problem faced in this work is to classify instances, GP has been set up to behave as a classification algorithm. A two-class classification problem, where there are only two classes, is typically less complex than a multiclass classification, where the program must classify more than two-classes. In addition, using GP for two-class classification benefits from the use of boolean functions (see the operators in Table 3). In order to optimize the effectiveness of GP regarding the multi-class classification, we have slightly modified the detection algorithm. In the following section we give details on this modification.

4.3.1. Modification of the GP algorithm

Algorithm 1 Return the class of *trace*

```
output ← Program_0 (trace)
if output is 0 then
   0
end if
output ← Program_1 (trace)
if output is 0 then
   1
end if
output ← Program_2 (trace)
if output is 0 then
   2
end if
output ← Program_3 (trace)
if output is 0 then
   3
end if4
```

In the multiclass scenario there are five different classes: normal (label 0), and each of the four different attacks (labels 1, 2, 3 and 4). In order to reduce the complexity of the problem, the five-class problem is translated into four two-class problems following the idea presented in [31]. Thus, four different programs were employed, each one specialised in detecting one specific class from the remainder, but which are intended to be executed together (thus, it is a unique intrusion detection module):

- Program_0: determines whether the trace is class 0 (it returns 0) or 1,2,3 or 4 (it returns 1).
- Program_1: determines whether the trace is class 1 (it returns 0) or 2,3 or 4 (it returns 1).
- Program_2: determines whether the trace is class 2 (it returns 0) or 3 or 4 (it returns 1).
- Program_3: determines whether the trace is class 3 (it returns 0) or 4 (it returns 1).

Programs may return 0 (if they detect the specific class) or 1 (the trace belongs to any other of the remainder classes). At first, the Program_0 is run, if it outputs a 0, a 0 is returned, otherwise the Program_1 is executed. If this outputs a 0, that means that the trace belongs to class 1, so a 1 is returned, otherwise, the Program_2 is executed, and so on (see the Algorithm 1) .

4.3.2. Performance Comparison Metrics

As in the multi-classification problem, the dataset is composed of traces of four different attacks, it does not make sense to talk about a single attack prevalence. Therefore, in multi-class problems the *intrusion detection capability* (Cid) is not an appropriate metric (as originally presented in [29]). To compare the classification algorithms we use the *false alarm* rate (FA), the *detection rate* (DR) and the *classification error* (CE). In a multiclass scenario, the *detection rate* and the *false alarm* rate are computed differently from the two-class case (Equations 1).

More precisely these metrics are calculated using composed probabilities and given by the equations below.

$$DR' = \frac{\sum_{i=1}^{n} P(A_i \& I_i)}{\sum_{i=0}^{n} \sum_{j=1}^{n} P(A_i \& I_j)}$$

$$FA' = \sum_{i=1}^{n} \sum_{j=0, j \neq i}^{n} P(A_i \& I_j)$$

Table 2: Contingency matrix showing the joint probabilities for a multi-class classifier. Columns (I_i) represent intrusion events of type i (0 denotes a normal event) and rows (A_j) correspond to the alarms of type i generated by the IDS (0 denotes the absence of an alarm)

		Intrusion			
		I_0	I_1	...	I_n
IDS	A_0	$P(A_0\&I_0)$	$P(A_0\&I_1)$	...	$P(A_0\&I_n)$
	A_1	$P(A_1\&I_0)$	$P(A_1\&I_1)$	...	$P(A_1\&I_n)$
	...	...	...	...	...
	A_n	$P(A_n\&I_0)$	$P(A_n\&I_1)$	...	$P(A_n\&I_n)$

In these equations, each $P(A_i\&I_j)$ represent the number of traces of the class j classified as i by the detector (thus, when $i = j$, is a correctly classified trace), divided by the total number of traces (n). Table 2 shows a description of contingency matrix for this case.

4.4. Algorithmic Technical Details

Our analysis is an extension of a previous work [7], where five different classifiers were used to detect malicious activity. In that work, it was stated that the best classifier was the Support Vector Machine (SVM), which uses Lagrangian methods to minimise a regularized function of the empirical classification error. In this work, we extend the comparison with a new classifier, based on Genetic Programming. In addition, we study new different simulated environments (with a lower prevalence of attacks) and use new performance comparison metrics (i.e., the Cid).

In order to tune the classification models, we have performed 10-fold cross validation [32] on the training datasets, which were created with random sampling. For each of the 10 folds we selected $1/10^{th}$ of the dataset for evaluation and the remaining for training. Finally, we use the selected parameters for each classification algorithm to train each model.

More precisely, for the MLP we tuned three parameters, i.e., the learning rate (η) and the number of iterations (T) used in the stochastic gradient descent optimization as well as the number of hidden units (nh). We kept nh equal to 0 and selected the appropriate η among values that range between 0.0001 and 0.1 with step 0.1 and the appropriate T was selected among 10, 100, 500 and 1000. After selecting the appropriate η and the appropriate T, we examined various values in order to select the appropriate nh. We selected the best among 10, 20, 40, 60, 80, 100, 120, 140, 160 and 320. For the *Linear* model we used the MLP model with no hidden units.

For the GMM, we tuned three parameters, i.e., the threshold (θ), the number of iterations (T) and the number of Gaussian Mixtures (ng). Keeping ng equal to 20, we selected the θ among values that range between 0.0001 and 0.1 with range 0.1 and the most suitable T among 25, 100, 500 and 1000. Finally, in order to select ng, after selecting the appropriate θ and the appropriate T we selected the best value for ng among 10, 20, 40, 60, 80, 100, 120, 140, 160 and 320.

For the SVM algorithm we tuned two parameters, i.e., the standard deviation (σ) for the gaussian kernel and the regularisation parameter c which represents the trade-off between the size of the margin and the number of misclassified examples. For the selection of the appropriate combination of σ and c, we examined various values for the σ $(1, 10, 100, 1000)$ and the c $(1, 10, 100, 1000)$ and selected the best.

For the Genetic Programming algorithm we tuned the following parameters:

- *Crossover rate:* Percentage of individuals on the population to be crossed. We examined any possible decimal value between 0 and 100.
- *Mutation rate:* Percentage of individuals on the population to be mutated. We have investigated possible values between 0 and 50.
- *Size of the population:* Number of individuals in the population. Values are restricted to be greater than 500 and lower than 1300.
- *Number of generations:* Normally, the larger the better, but if the evolution remains stagnant, using high values may be inefficient. The range of possible values given is from 60 to 200.
- *Tournament size:* It is the number of individuals selected to perform the tournament selection (see [21] for more information). We accept integer values between 3 and 8.

We have run 140 experiments, with 140 different configurations. Table 3 shows the parameters obtained that finally were used in the experiments. and the list of functions (internal nodes of the trees) used. The list of terminals, described in Table 1, corresponds to the set of features in the dataset. As

Table 3: GP parameters and operators used in the experiments

Number of generations	193
Size of population	942
Size of tournament	7
Crossover rate	13%
Mutation rate	49%
Operators	ADD, AND, DIV, GREATER, LEAST, MULT, OR, MAX, MIN, ROTATE_LEFT, NOT

mentioned above, the fitness function is a critical component as it defines the way the individuals must evolve to solve the problem. The experiments were performed using two different fitness functions, i.e. minimising the *classification error* (CE) and maximising the *intrusion detection capability* (Cid).

5. Results

SVM was the best classifier of the five studied previously (excluding GP) [7]. Therefore, the comparison regarding the two-class classification is made between SVM and the new classifier we are analysing, i.e., the one based on GP. Figure 2 shows the comparison of SVM with the two GP individuals obtained using the original dataset with an attack prevalence of 80%. From now on, we denote as GP1 to the model obtained trying to maximize the Cid, and GP2 the model obtained minimizing the *classification error* (CE). GP improves the results obtained with SVM, as the *classification error* (CE) of both models are lower than the SVM and the Cid is quite higher.

When the number of attacks in the dataset is reduced, we can see in both Figure 3 (4% of attack prevalence) and Figure 4 (1% of attack prevalence) that the *classification error* (CE) is lower when the GP algorithm is employed. Regarding the Cid when using the dataset with a 1 % of prevalence (Figure 4), the Cid is better for both GP models. The GP1 model maximizes the Cid, so it obviously achieves the best value (0.29 approximately). The GP2 model accomplishes the lower *classification error* (CE) in all cases. However, as we can observe in Figure 3, regarding the Cid the SVM model has slightly better performance than the GP algorithm.

Regarding the multiclass dataset, as stated in the Section 4.3 the comparison is done in terms of *detection rate* (DR), the *false alarm rate* (FA) and

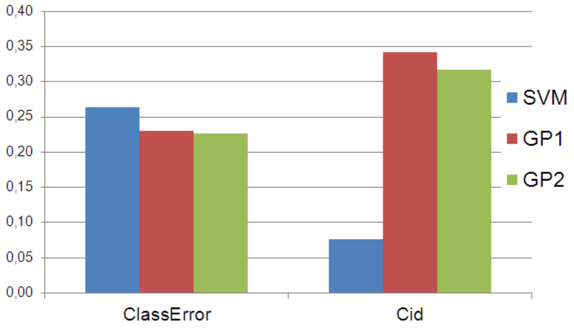

Figure 2: Results achieved for the original two-class dataset with an attack prevalence of 80%

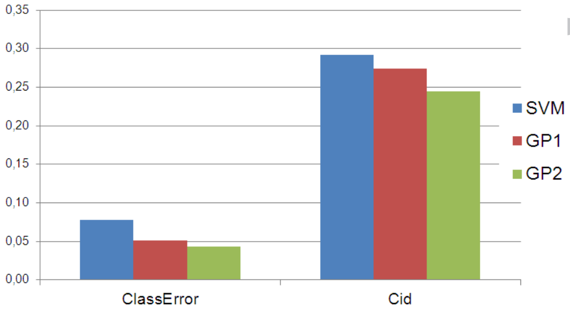

Figure 3: Results achieved for the modified two-class dataset with an attack prevalence of 4%

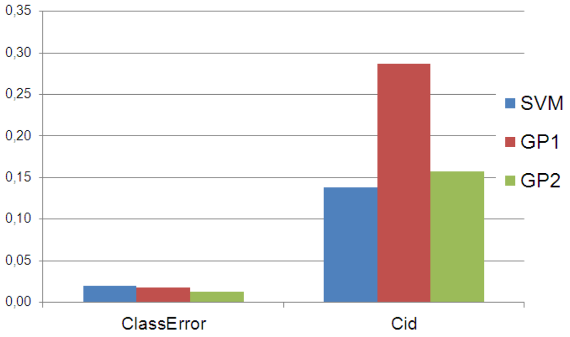

Figure 4: Results achieved for the modified two-class dataset with an attack prevalence of 1%

Table 4: False Alarm rate (FA), Detection Rate (DR) and Classification Error (CE) incurred by each classifier using the multiclass dataset.

	FA	DR	CE
GMM	0.08	**0.77**	0.21
MLP	**0.05**	**0.77**	**0.18**
LINEAR	**0.05**	0.76	0.19
NAIVE	0.14	0.72	0.26
SVM	0.06	**0.77**	0.19
GP	**0.05**	0.73	0.21

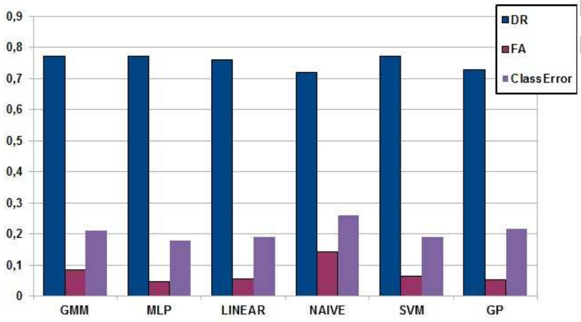

Figure 5: Graphical comparison of the classifiers regarding the multiclass dataset

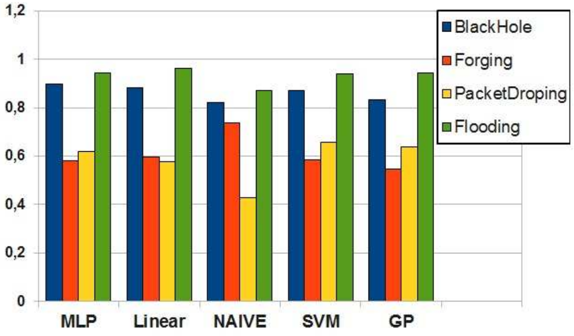

Figure 6: Detection rates of each attack by each classifier regarding the multiclass dataset

the *classification error* (CE). Figure 5 and Table 4 show the performance results for all the classifiers studied. It can be seen that the best performance is achieved by the MultiLayer Perceptron (MLP) and the SVM, with nearly equal performance. The GP classifier has a lower *detection rate* (DR), but also a lower *false alarm rate* (FA), so the *classification error* (CE) is quite high, as the multiclass dataset has a high prevalence (80%). Therefore, if the GP algorithm detects an attack, it is known with a higher certainty that this attack is actually happening, since the probability of giving false alarms (FA) is one of the lowest for all the classifiers. Figure 6 shows the *detection rate* (DR) for all the classifiers and for each attack. The attack with the highest detection rate for the GP algorithm is the Flooding attack. It can be seen that, although GP is not a good multiclass classifier, it has better performance than other classifiers for some attacks. For instance, it detects a Forging attack better than the MLP, the Linear and Naïve Bayes classifier, and presents almost equal performance with the SVM classifier. This implies that the technique of dividing a multi-classification problem into several two-class classification problems may work under certain conditions or attacks. This is a further line to investigate in the future.

6. Conclusions

Detecting malicious activities in MANETs is a complex task because of the inherent features of these networks, such as the mobility of the nodes, the lack of a fixed architecture as well as the severe resource constraints. There is an urgent need to safeguard these communication networks and to propose efficient mechanisms in order to detect malicious behaviour.

In this article we provide a comparison of the effectiveness of different classifiers that can be employed as intrusion detection algorithms in MANETs. Results show that Genetic Programming may be a good paradigm to use when the goal is just to detect an intruder, although if the objective is to indicate which is the particular attack launched then it is better to use a SVM classifier. The evaluation of the classifiers is performed considering that the intrusion detection process is completely distributed and each node of the network hosts an independent intrusion detection agent. As future work we will examine which network architecture is more efficient regarding the placement of the intrusion detection agents in the network. Additionally, we plan to analyse new approaches to detect intrusions, such as the promising Artificial Immune Systems (AIS) [33, 34].

7. Acknowledgements

This work has been partially supported by the Marie Curie IEF, project "PPIDR: Privacy-Preserving Intrusion Detection and Response in Wireless Communications", grant number 252323, and also by the Comunidad de Madrid and Carlos III University of Madrid, Project EVADIR CCG10-UC3M /TIC-5570.

References

[1] B.-C. Cheng, R.-Y. Tseng, A Context Adaptive Intrusion Detection System for MANET, Computer Communications 34 (2011) 310–318.

[2] F. Anjum, P. Mouchtaris, Security for Wireless Ad Hoc Networks, Wiley-Interscience, 2007.

[3] M. Al-Shurman, S.-M. Yoo, S. Park, Black-hole Attack in Mobile Ad Hoc Networks, in: Proceedings of the 42nd Annual Southeast Regional Conference, ACM-SE 42, ACM, New York, NY, USA, 2004, pp. 96–97.

[4] P. Ning, K. Sun, How to Misuse AODV: a Case Study of Insider Attacks Against Mobile Ad-hoc Routing Protocols, Ad Hoc Networks 3 (2005) 795–819.

[5] D. Djenouri, O. Mahmoudi, M. Bouamama, D. Llewellyn-Jones, M. Merabti, On Securing MANET Routing Protocol Against Control Packet Dropping, in: Proceedings of the International Conference on Pervasive Services, IEEE Computer Society, CA, USA, 2007, pp. 100–108.

[6] P. Yi, Y. F. Hou, Y. Zhong, S. Zhang, Z. Dai, Flooding Attack and Defence in Ad hoc Networks, Journal of Systems Engineering and Electronics 17 (2006) 410 – 416.

[7] A. Mitrokotsa, M. Tsagkaris, C. Douligeris, Intrusion Detection in Mobile Ad Hoc Networks Using Classification Algorithms, in: Proceedings of the Seventh Annual Mediterranean Ad Hoc Networking Workshop, Springer, Palma de Mallorca, Spain, 2008, pp. 133–144.

[8] R. Bace, P. Mell, NIST Special Publication on Intrusion Detection Systems, Technical Report, National Institute of Standards and Technologies, 2001.

[9] S. Sen, J. A. Clark, Intrusion Detection in Mobile Ad Hoc Networks, Springer, pp. 427–454.

[10] Y. Zhang, W. Lee, Intrusion Detection in Wireless Ad-hoc Networks, in: Proceedings of the 6th annual international conference on Mobile computing and networking, MobiCom, ACM, New York, NY, USA, 2000, pp. 275–283.

[11] Y. Zhang, W. Lee, Y. Huang, Intrusion Detection Techniques for Mobile Wireless Networks, Wireless Networks 9 (2003) 545–556.

[12] Y. Huang, W. Lee, A Cooperative Intrusion Detection System for Ad hoc Networks, in: Proceedings of the 1st ACM Workshop on Security of Ad hoc and Sensor Networks, SASN '03, ACM, New York, NY, USA, 2003, pp. 135–147.

[13] Y.-a. Huang, W. Lee, Attack Analysis and Detection for Ad Hoc Routing Protocols, in: E. Jonsson, A. Valdes, M. Almgren (Eds.), Proceedings of the Recent Advances in Intrusion Detection, volume 3224 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2004, pp. 125–145.

[14] C. Perkins, RFC 3561: Ad hoc On-Demand Distance Vector (AODV) Routing, 2003. Status: EXPERIMENTAL.

[15] O. Kachirski, R. Guha, Effective Intrusion Detection Using Multiple Sensors in Wireless Ad hoc Networks, in: Proceedings of the 36th Annual Hawaii International Conference on System Sciences, IEEE, 2003, pp. 8–16.

[16] B. Sun, K. Wu, U. Pooch, Zone-based Intrusion Detection System for Mobile Ad hoc Networks, International Journal of Ad Hoc and Sensor Wireless Networks 2 (2006).

[17] M.-Y. Su, Prevention of Selective Black Hole Attacks on Mobile Ad hoc Networks Through Intrusion Detection Systems, Computer Communications 34 (2011) 107–117.

[18] S. Sen, J. A. Clark, A Grammatical Evolution Approach to Intrusion Detection on Mobile Ad hoc Networks, in: Proceedings of the Second

ACM Conference on Wireless Network Security, WiSec'09, ACM, NY, USA, 2009, pp. 95–102.

[19] S. Sen, J. A. Clark, J. E. Tapiador, Power-Aware Intrusion Detection in Mobile Ad Hoc Networks, in: Proceeding of the Ad Hoc Networks, volume 28 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Springer Berlin Heidelberg, 2010, pp. 224–239.

[20] C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery 2 (1998) 121–167.

[21] J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems), The MIT Press, 1992.

[22] G. S. Mark Crosbie, Applying Genetic Programming to Intrusion Detection, in: Proceedings of the AAAI Fall Symposium on Artificial Intelligence, COAST Publications, West Lafayette, IN 47907-1398, 1995, pp. 1–8.

[23] A. Orfila, J. M. Estevez-Tapiador, A. Ribagorda, Evolving High-Speed, Easy-to-Understand Network Intrusion Detection Rules with Genetic Programming, in: Proceedings of the EvoWorkshops on Applications of Evolutionary Computations, volume 5484 of *Lectures Notes in Computer Science*, Springer Berlin / Heidelberg, 2009, pp. 93–98.

[24] G. Folino, C. Pizzuti, G. Spezzano, GP Ensemble for Distributed Intrusion Detection Systems, Lecture Notes in Computer Science 3686 (2005) 54.

[25] B. Kavitha, D. S. Karthikeyan, P. S. Maybell, An ensemble design of intrusion detection system for handling uncertainty using Neutrosophic Logic Classifier, Knowledge-Based Systems 28 (2012) 88 – 96.

[26] S. Pastrana, A. Orfila, A. Ribagorda, A Functional Framework to Evade Network IDS, in: Proceedings of the 44th Hawaii International Conference on System Sciences, IEEE, Koloa, Hawaii, USA, 2011, pp. 1–10.

[27] X. Zeng, R. Bagrodia, M. Gerla, GloMoSim: a Library for Parallel Simulation of Large-scale Wireless Neworks, in: Proceedings of the

12th Workshop on Parallel and Distributed Simulations, IEEE, Banff, Alberta, Canada, 1998, pp. 154–161.

[28] B. Sun, Intrusion Detection in Mobile Ad Hoc Networks, Ph.D. thesis, Department of Computer Science, Texas University, 2004.

[29] G. Gu, P. Fogla, D. Dagon, W. Lee, B. Skorić, Measuring Intrusion Detection Capability: an Information-theoretic Approach, in: Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, ACM, New York, NY, USA, 2006, pp. 90–101.

[30] J. Blasco, A. Orfila, A. Ribagorda, Improving Network Intrusion Detection by Means of Domain-Aware Genetic Programming, in: Proceedings of the 2010 International Conference on Availability, Reliability and Security, IEEE, 2010, pp. 327–332.

[31] J. Fürnkranz, Round Robin Classification, Journal of Machine Learning Research 2 (2002) 721–747.

[32] R. Kohavi, A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection, in: Proceedings of the International Joint Conferences on Artificial Intelligence, volume 2, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995, pp. 1137–1145.

[33] J. Yang, X. Liu, T. Li, G. Liang, S. Liu, Distributed agents model for intrusion detection based on AIS, Knowledge-Based Systems 22 (2009) 115 – 119.

[34] M. Gong, J. Zhang, J. Ma, L. Jiao, An Efficient Negative Selection Algorithm with Further Training for Anomaly Detection, Knowledge-Based Systems 30 (2012) 185–191.