IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 10, 2019, accepted June 25, 2019, date of publication July 12, 2019, date of current version August 2, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2928414*

# Security Agent Location in the Internet of Things

**CHRISTIANA IOANNOU AND VASOS VASSILIOU, (Senior Member, IEEE)**
Research Center on Interactive Media, Smart Systems and Emerging Technologies (RISE), 1011 Nicosia, Cyprus
Department of Computer Science, University of Cyprus, 2109 Nicosia, Cyprus

Corresponding author: Vasos Vassiliou (vasosv@cs.ucy.ac.cy)

This work was supported in part by the EU's H2020 research and innovation programme under Grant agreement 739578, in part by the Government of the Republic of Cyprus through the Directorate General for European Programmes, Coordination and Development, and in part by the the University of Cyprus under the Internal Project "SMIDS: Detecting Malicious Interventions in Wireless Sensor Networks and the Internet of Things".

**ABSTRACT** The Internet of Things (IoT) provides the ability to extend the Internet into devices and everyday objects, in a way that they are uniquely addressable. Sensors, actuators, as well as everyday devices and objects, such as cellphones, cars, and homes, are interconnected and form a network that can be accessed, monitored, and controlled remotely. Security is an important subject in the IoT networks since the devices and the networks can be used as a means of invading the users' privacy. The current work examines the issue of security agent location using indicative intrusion detection techniques for network layer attacks. We analyze the methodology, operation, as well as the complexity of each technique. Through the extensive implementation and experimentation, we are able to conclude that the local security agents have the same performance results with centralized and decentralized approaches, but with negligible overhead. As such, they are useful when internal network communication, or network augmentation with monitoring nodes, is not feasible.

**INDEX TERMS** Internet of Things, intrusion detection systems, security agents, anomaly detection, logistic regression, support vector machines.

## I. INTRODUCTION

The trend to socially interact through the Internet has led to a new technological era in the field of network communications and services and shifted the attention towards connecting everyday objects. This mode of connectivity has motivated people to access sensor and embedded device data via the Internet, thus creating the Internet of Things (IoT). The Internet of Things is essentially an extension of the Internet into devices and everyday objects, which can be accessed through the Internet in a uniquely addressable way. The ease of communication with and between the things has boosted the development of many applications that can provide new services to citizens, companies, and public administration [1], [2]. Smart homes, Smart Cities, Smart Critical infrastructures are some of the IoT examples that can make life easier. Networks of this type inevitably attract people with malicious intent, who aim in disrupting their functionality by any means possible.

The Things in the IoT inherit wireless sensor security vulnerabilities and at the same time are connected to the largest untrusted network, the Internet, making them more vulnerable and appealing for malicious interventions. Smart City applications can draw attention to perpetrators as the IoT applications become massive databases for the city. In traditional networks, the end users have some control on the security measures to adopt and they can avoid getting hacked by updating their software and choosing the applications to be executed. In Smart City applications the citizens cannot control the security level of the underlying IoT. They may be ignorant of its existence, but the perpetrators can definitely violate it and, as a result, violate the citizens' privacy.

The first line of defense in computer systems and networks is the use of protocols and mechanisms that try to maintain integrity, confidentiality and authentication, thus preventing known malicious intruders from entering the network [3]. Intrusion Detection is considered to be the second line of security defense [3]. The detection layer is responsible for recognizing attacks, which have managed to breach the first line of security defense and/or which try to disrupt the network's availability by acting within the network or the node.

The current work examines the issue of the placement of security agents (SA) within the IoT network. Our contributions are: (a) the specification of the main ways to place security agents and representative techniques for each; (b) the

The associate editor coordinating the review of this manuscript and approving it for publication was Pietro Savazzi.

design of a framework in which the techniques are trained and evaluated using both benign and malicious node activity; (c) the creation of the node activity using network layer attacks implemented according to their most accepted definitions; (d) the deployment of the attacks against a simple routing protocol, to remove any artifacts resulting from complex implementations based on standards; (e) the detailed analysis of three different anomaly IDS techniques; and (f) the extensive evaluation of results, using multiple performance metrics.

The rest of the paper is structured as follows: Section II presents common anomaly detection techniques and elaborates on the techniques chosen to be evaluated. Section III presents the steps taken to implement routing attacks, execute them, and monitor sensor activity. Section IV, V, and VI show the operations, training, and the evaluation results for each detection technique. Section VII contains our conclusions.

## II. BACKGROUND AND RELATED WORK

There are two basic intrusion detection techniques: anomaly detection and pattern-based detection. Pattern detection, also known as signature-based, detects attacks which have been previously identified. For each attack detected, its unique pattern and/or signature is computed, stored and used to compare with real-time patterns. Anomaly detection aims in detecting known and novel attacks by identifying abnormal activity. Any deviation from normal activity is considered to be malicious. Both detection techniques bear advantages and disadvantages. The pattern detection technique has the advantage of not raising many false alarms, since all the alarms are based on known attacks. Nevertheless, to be able to recognize a new attack, the attack first needs to happen, which means that the unfortunate and unsuspecting networks will be damaged. The anomaly detection technique has the advantage of detecting novel attacks more easily by recognizing abnormal activity.

Abnormal activity can be the result of either malicious intervention or node malfunction. There are two stages of operation in each anomaly detection technique; the training and the deployment stage. At the training stage, normal activity is defined. Defining normal activity is crucial in minimizing false alarm rates and increasing detection rates [4]. One key factor for determining normal activity is to take into consideration both normal and abnormal conditions [4]. An Intrusion Detection System (IDS) anomaly detection agent is equipped with the knowledge of what normal activity is and a tolerance threshold. At the deployment stage the IDS agent monitors node and/or network activity. If the activity exceeds a predetermined threshold or does not comply with the definition of normal activity, it is marked as abnormal and an alarm is raised.

Anomaly detection mechanisms can be classified based on many things. One classification is based on the source of the audit data, which may be the node, the network, or both. Another classification is based on the method used for profiling normal activity and capturing abnormalities. There are

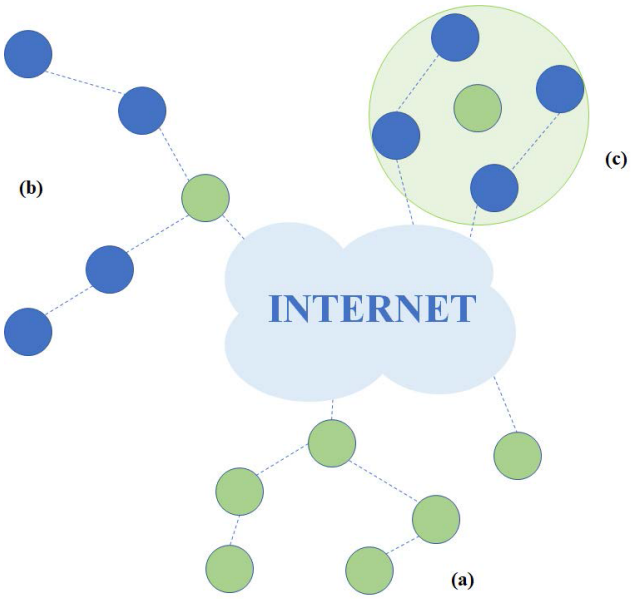


**FIGURE 1.** Placement of Security Agents (SA) (a) BLR using local SA, (b) SVM using global SA, and (c) threshold based decentralized SA.

numerous surveys on the topic [5]–[7], so we will not expand here. It suffices to recognize that anomaly IDS are roughly classified as Machine Learning-based, Statistical Analysis-based, and Rule/Threshold-based.

The classification we consider in this work relates to the location of the security agents and the way they operate. There are three key ways to place an intrusion detection agent: locally/distributed, globally/centralized, and decentralized [6]. Local security agents are installed in every IoT object and are responsible both for monitoring node behavior and for detecting node malfunction or malicious operation (see Fig. 1a). In certain cases, local agents can also be assigned the role of monitoring neighboring traffic and detecting malicious neighbors, thus becoming distributed security agents. The concern with local security agents is the computational and memory overhead they may impose [6]. A more centralized approach is having a global security agent installed within a central node, the Sink or Gateway node, that is responsible for monitoring all network traffic and detecting attacks (see Fig. 1b). There are two major drawbacks in having a global agent: it creates additional traffic to transmit node status reports and in case of an attack the global agent may have a difficulty monitoring the network traffic, as most of the traffic may not reach their destination [6], [8]. Decentralized security agents, place multiple agents within the network to monitor clusters of nodes (see Fig. 1c). The decentralized approach is a combination between distributed agents and global agents aimed to avoid the drawbacks of both. According to [6], the advantage of this approach is that each decentralized security agent can be customized to each cluster of nodes it monitors and it can set up its own set of rules to achieve better results. The disadvantage of this ability to customize the operation is the configuration overhead.

For the purpose of this work, we have chosen a representative solution for each placement strategy. For the decentralized solution we use the rule-based approach proposed by R. da Silva et al. in [9]. For the centralized approach, we analyze an anomaly detection technique using a C-Class support vector machine (SVM) model. This is based on the work of Kaplantzis et al. in [10], but extended with both malicious and benign activity, for a more precise definition of abnormalities, as suggested by [11]. The complexity and overhead of this technique places the security agent at a powerful node able to monitor, or receive, the whole network activity. The third method to be evaluated is the Binary Logistic Regression statistical model, as suggested by Ioannou et al. [12]. The computation and complexity of the model allows to analyze a local security agent approach, in which every node is also a security agent that bases its detection on local node activity.

The rest of the section, presents details for each of the three detection techniques.

### A. LOCAL IDS USING BINARY LOGISTIC REGRESSION

Binary Logistic Regression (BLR) allows the analysis and prediction of a dichotomous outcome. It is a statistical method that typically predicts a binary dependent variable based on a set of independent variables. Each independent variable can be continuous, discrete, categorical, or binary. BLR Logistic regression is also being applied in health care and financial and economic models to provide prognosis [13]. Logistic regression has also being used in [14] to evaluate Wireless Sensor Network (WSN) data reliability. In [15] they used Logistic Regression (LR) to detect patient anomalies and sensor faults in medical WSNs. The work in [16] is closely related to ours as they use BLR to detect malicious system calls within a personal computer. They monitored system calls invoked by both benign and malicious applications in a Windows operating system.

### B. GLOBAL IDS USING SUPPORT VECTOR MACHINES

Support Vector Machine (SVM) is a class of machine learning algorithms that can create a binary classification model out of complex highly non-linear problems [10]. An SVM performs classification tasks by constructing hyperplanes in a multidimensional space that separates cases of different class labels. SVM has been used to detect possible intruders in the network such as the work in [10] and to detect outliers in the data [17]. In both [10] and [17], they use one-class SVMs as they trained their system using only the normal activity of the network and sensing data. They support that by only using benign activity, an unknown attack can still be accurately detected.

In their work, a centralized node (the Sink) is responsible for intrusion detection including data processing and anomaly detection tasks. Their scheme of a centralized node detection system was created so as not to impose memory and power overhead on the constrained nodes.

### C. DECENTRALIZED IDS USING THRESHOLDS

The authors in [18], proposed monitoring a WSN network using performance parameters between source and destination nodes, such as packet loss, signal to noise ratio, bit error rate, number of hops, and energy consumption. The work proposed in [19], detects insider attacks by monitoring packet dropping rate, packet sending rate, forwarding delay time, and sensor readings. The packet dropping rate requires buffering the recently sent packets. The packet sending rate counts the number of packets sent for a specific time lapsed. Forwarding delay time computes the difference in time between the time a packet takes to be transmitted to node $x$ and the time node $x$ takes to forward the packet. The authors in [20] proposed an IDS system based on an average received power and average packet arrival rate.

In [21], they implement an IDS with four modules. Each sensor sends status messages to the server. The heartbeat module periodically sends status messages with no specific data information to the server to acknowledge its existence. Lack of status messages from a node can be classified as a disconnected or faulty node. For the heartbeat module, they leveraged the data messages that the nodes were sending to the server. The movement module is triggered when the sensor senses a movement. A threshold is applied at the server to investigate whether the node has been physically moved or sensed a movement. The Carrier Sense Time (CST) module sends alert messages to the server if a certain threshold for the CST has been exceeded indicating the presence of a security threat called jamming. The last IDS module is the OTAP module in which it is responsible to detect, using signatures, any unauthorized attempt for reprogramming the node. It sends alarm messages to the server for further investigation.

The work in [22] proposes various metrics to detect DoS attacks in WSNs. At predefined time intervals, cluster head nodes request information from nodes within the network, either from a subset on nodes within clusters of nodes or specific nodes, and evaluate them. The information requested include power consumption, packets received and sent. The parameters are then formed into metric values and thresholds are applied to determine whether there is an intrusion.

In [9], the authors propose applying rules to detect possible attacks within the network. Each rule is customized to detect specific attacks. When multiple rules are violated, their IDS raises an alarm. They define five rules in which they can detect multiple attacks when they exceed a limit. The *interval rule* monitors the time passed between the reception of two consecutive messages within the allowed limits. At the *retransmission rule*, the monitoring node listens to the packet sent from a node and received by another node within its vicinity. If the relay node does not forward the packet received an alarm is raised. The *integrity rule* monitors messages' payload that needs to be the same from the origin to the destination. The *delay rule* monitors retransmission time, the repetition rule monitors the number of times a message is retransmitted. The *radio transmission range rule* monitors

the messages' origin. If the message received is not from a neighboring node then an alarm is raised. The last rule is the *jamming rule* that monitors the number of collisions associated with a message.

The IDS agents are distributed among the network and are responsible to monitor and detect possible attacks. Their method requires that the constrained nodes are monitored by at least one monitoring node. A monitoring node is responsible to eavesdrop their neighbors and apply detection rules.

## III. METHODOLOGY

To evaluate the different IDS techniques we executed the following four steps:

1) Definition and Implementation of Attacks
2) Data Collection Methodology
3) IDS operation
4) Evaluation

In this section we will explain items 1), 2) and 4). Details of each IDS technique (item 3) are presented in Sections IV, V, and VI for the Local, Centralized, and Decentralized approaches respectively.

For most of the components in the current work, we used the Contiki operating system [23] and the associated Cooja simulator [24]. We created our own routing protocol; implemented the attacks, based on the most common definition of them; collected the data using a specially created monitoring tool; simulated the attacks; retrieved the experimental data for training purposes; and, finally, implemented the IDS techniques.

### A. WEIGHED SHORTEST PATH ROUTING PROTOCOL

The attacks were created over an in-house implemented simple routing protocol called WSP, which is based on the widest-shortest path concept. Numerous routing protocols have been proposed to be used for the IoT, the most popular being the Routing Protocol for Low Power and Lossy Networks (RPL) over 6LowPAN (Low power Wireless Personal Area Networks) [1], [25], [26]. To evaluate the anomaly techniques independent of the routing protocol constraints we chose to implement the attacks using a simple routing protocol with minimal signaling overhead.

WSP takes into consideration the distance of the sensor node from the Sink node and the signal strength of its neighbors. The protocol requires that each sensor node has a neighbor table, which is created when the network is formed and is updated periodically or after an event. A table entry includes the neighbor's node id, the number of hops to the sink, and the last measured RSSI value to that neighbor. Once a node is activated, it transmits broadcast control messages, announcing its existence. Unless the node is the Sink, it advertises a hop count of $-1$, up until it figures out its location within the routing tree topology. The Sink advertises a hop count of 0. The Sink's neighbors are the first nodes that update their hop count to 1 hop away from the Sink and broadcast it. Their neighboring nodes update their tables accordingly, and broadcast their new hop count set to 2. The process is repeated up until all nodes are connected. A sensor node chooses a relay node, initially based on the distance from the Sink and then according to the highest RSSI.

### B. NETWORK LAYER ATTACKS

Routing attacks aim in exploiting routing protocol vulnerabilities to disrupt the network's flow by diverting traffic towards the compromised node making it difficult for the network to accomplish its intended goal. The compromised node may alter, redirect, or drop the packets received, thus changing the network operation. The current work focuses on three routing attacks, the Selective Forward, the Blackhole and the Sinkhole attacks.

The Selective Forward (SF) attack, as the name implies, selectively chooses which packets to forward and which ones to drop. In the current work, the selection of the packets is done randomly; either by a predefined probability, named Forwarding Ratio (FR) or by blocking specific neighboring node's packets for a specific time interval, named Blocking Node (BN). The Blackhole (BH) attack maliciously advertises that the malicious node is one hop away from the Sink. The BH attack is used in combination with SF attack and with both dropping selection methods.

The Sinkhole attack advertises itself as the Sink and lures traffic from all the network, not just the neighboring nodes, towards itself. A Sinkhole attack can then either tamper with routing packets, spoof or replay route messages, or even transmit false report attacks to constitute the compromised node as a more attractive path to forward packets [27]–[29]. The victim nodes transmit their packets toward the malicious node and their packets are essentially lost [27]. The best location to create a Sinkhole attack is near the Sink where it is easier to ''fool'' neighboring nodes and get all the traffic that was destined for the Sink [28]. It was shown that depending on the network topology, even when the malicious node is located further away from the Sink node, it can still deny the Sink the ability to receive packets from large portions of the network [8].

### C. DATA COLLECTION

Data collection from each node was done using the Remote Monitoring Tool (RMT) described in [30]. RMT is able to collect data from any layer of the protocol stack and also collect information on CPU activity and power consumption. For this work, we are using data from the network layer that includes *data packets received*, *data packets sent*, *number of packets forwarded*, *number of packets dropped*, and *number of announcements received* (see Table 1).

### D. EXPERIMENTAL DATA

Each detection technique was trained and evaluated over the same network topology. Fig. 2 shows the network topology we used. The Sink node (shown in a dark color) is in the middle of a $5 \times 5$ grid. The numbers in each node show the distance, in hops, of the node to the Sink. The Sink has four

**TABLE 1.** Network layer parameters.

| WSP | |
|---|---|
| Data Packets Received | Data Packets Sent |
| Packets Forwarded | Packets Dropped |
| Announcements Received | |

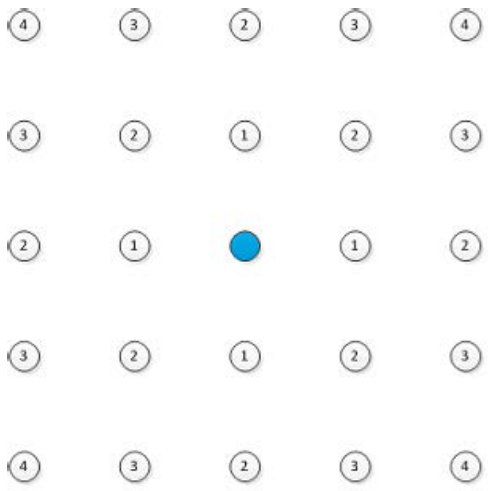


**FIGURE 2.** Network topology: Sink in the middle of the grid.

neighboring nodes and the farthest nodes are four hops away from the Sink.

For the Local and Centralized methods, two types of scenarios were created to gather data and evaluate the detection mechanism. In a benign scenario all nodes within the network are executing a benign application. For each topology we execute 10 benign simulations each with a different random seed. In a malicious scenario one node within the network is executing a malicious application. For each network topology we executed 24 malicious scenarios for the Selective Forward attack, 24 malicious scenarios for the Blackhole attack, and 24 malicious scenarios for the Sinkhole attack.

The scenarios were evaluated within a 15-minute experimental time. The nodes started transmitting data after the first 2 minutes of experimental time. The 2-minute window was set to allow the sensor nodes to be connected and reach a steady state. The RMT monitoring time interval was set to 30 seconds, thus the data retrieved in the 15-minute experimental time was for 26 monitoring periods. Only the first 25 monitoring time intervals were used, as experimental time expired before certain nodes had the chance to compute the last monitoring time interval.

In both benign and malicious applications, the data rate was set to one (1) packet per second (for an effective data rate of 384 bps). Every node was transmitting 30 packets per monitoring time interval and if it was a relay node, it was also forwarding the packets received by its neighboring nodes. At the end of the experimental time, the data gathered for each malicious scenario at the WSP layer was 25 monitoring periods from 24 constrained nodes.

**TABLE 2.** Confusion matrix.

| | | Diagnosis | |
|---|---|---|---|
| | | Viral | Benign |
| True Condition | Viral | True Positive | False Negative |
| | Benign | False Positive | True Negative |

The data per monitoring time was used as input for evaluating the local and global detection methods described in the following sections. In the decentralized approach, the data that was used was based on monitoring traffic between neighboring nodes. Section VI describes in detail the steps taken to monitor data and detect abnormal traffic between neighbors.

### E. EVALUATION METRICS

The confusion matrix shown in Table 2 classifies the alarms into four value types. The True Positive/True Negative values are the correct predictions of the model. True Positive (TP) values are the number of viral node activities that the model has correctly identified. The same applies to True Negative (TN) values; when given a benign activity, the model correctly identifies it, and no alarm is raised. On the contrary, the False Positive/False Negative values show the misclassifications of the model. The False Positive (FP) values correspond to benign node activities that are detected by the model as malicious. The False Negative (FN) values are the malicious sensor activities that have not raised an alarm.

The Accuracy and Precision ratio (ACC) shows the ratio of correct classifications over the total classifications made [31]. The ACC is shown in (1)

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Recall, also known as True Positive Rate (TPR), shows the ratio of TP, the alarms that were raised by the malicious nodes, over the total number of malicious events.

$$Recall/TPR = \frac{TP}{TP + FN} \tag{2}$$

The Recall/TPR is usually accompanied with the Precision, or Positive Predictive Value (PPV), that shows the percentage of correctly identified malicious nodes within the network (see (3)) [32]. The Precision/PPV has also been used with the name Detection Rate in [25], [33].

$$Precision/PPV = \frac{TP}{TP + FP} \tag{3}$$

## IV. LOCAL IDS USING BINARY LOGISTIC REGRESSION

The current section describes a detection method based on local SA (see Fig. 1a). Binary Logistic Regression (BLR) is used to predict a binary dependent variable based on a set of independent variables. To avoid a high number of false alarm rates, by defining strict thresholds, we use BLR analysis that takes as input both benign and viral local node activity and outputs a profiling equation. The dependent variable to be predicted is whether the activity is caused by a viral node or

not. The independent variables are a set of network activity parameters (shown in Table 1).

There are two stages in this approach, the training and the evaluation stage. The training stage takes as input a set of sample node activity data and outputs a regression model. The training stage indicates which independent variables are the most significant to the regression model [16]. The evaluation stage tests the model against a different data set.

For the training set, the independent variables are used as gathered to avoid any computation overhead that may be required at the deployment stage. The dependent variable classifies the nature of the node activity, whether it was created by a malicious or a benign node. Probability P, as shown in (4), is the probability that the network activity is ''viral''.

$$P = \frac{e^{\alpha+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_n x_n}}{1+e^{\alpha+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_n x_n}} \tag{4}$$

where $n$ is the number of independent variables. Each $x_i$ represents the value of $i^{th}$ independent variable, $\alpha$ is constant, $\beta$ is the regression coefficient.

The challenge of logistic regression is to fit the mathematical data with the actual data such as to maximize the goodness of fit. The mathematical tool for computing the goodness of fit is called the maximum likelihood which is the condition probability of the dependent variable, given the set of independent variables $P\,(DependentVariable|X)$.

For this research the p-value was used to evaluate the independent variable significance. The p-value for $x_i$ is computed using generalized linear model (see (5)).

$$Y = \alpha + \beta_1 x_{1,1} + \beta_2 x_{2,2} + \cdots + \beta_n x_{i,j} \tag{5}$$

where $Y$ is a matrix with the independent variables. Each $X$ represents the value of the independent variable, $\alpha$ is a constant, $\beta$ is the regression coefficient.

The independent parameters chosen were those with a p-value more than 0.05. Once a set of important independent variables is determined by the p-value, equations (4) and (5) were used iteratively using the evaluation data to verify their importance.

### A. TRAINING STAGE

For the training set, the independent variables are used as gathered and averaged per node distance offline. To classify the vectors, whether they are taken from benign or malicious nodes, we include the dependent variable. The dependent variable has two values: 0, for ''benign'', and 1 for ''viral''. The classification process allows the logistic regression to evaluate the significance of each independent variable in identifying the nature of the activity.

We refer to the set of benign vectors as $B$ and the set of the viral vectors as $V$. The vector representation of benign and viral vectors is the following:

$$\mathbf{b}_i = \{b_{i,1}, b_{i,2}, \ldots, b_{i,j}\}^T \tag{6}$$

**TABLE 3. Training and evaluation data sets when sink is in the middle.**

| Attack | $j$ in $X_{training}$ | | | $j$ in $X_{evaluation}$ | | |
|---|---|---|---|---|---|---|
| | Benign | Viral | Total | Benign | Viral | Total |
| SF - BN | 80 | 80 | 160 | 20 | 20 | 40 |
| SF - FR | 80 | 80 | 160 | 20 | 20 | 40 |
| SF & BH - BN | 80 | 80 | 160 | 20 | 20 | 40 |
| SF & BH - FR | 80 | 80 | 160 | 20 | 20 | 40 |
| Sinkhole | 80 | 80 | 160 | 20 | 20 | 40 |

where $\mathbf{b}_{i,j}$ represents the local activity $j$ made by a benign node at time interval $i$.

$$\mathbf{v}_k = \{v_{k,1}, v_{k,i}, \ldots, v_{k,j}\}^T \tag{7}$$

where $\mathbf{v}_{k,j}$ represents the local activity $j$ made by viral node at the time interval $k$.

The local node activities for the $B$ are extracted from the benign scenario of each topology in which no compromised node is present. The local node activities for the $V$ are extracted from the corresponding malicious scenario where there exists a compromised node within the network.

$$\underline{T}_{training} = \{t_{i,1}, t_{i,2}, \ldots, t_{i,j}\} \tag{8}$$

$$\underline{U}_{training} = \{u_{k,1}, u_{k,2}, \ldots, u_{k,j}\} \tag{9}$$

We created two data sets: $T_{training}$ by using 80% of the activities from $B$ and $U_{training}$ by using 80% of the activities from $V$. The final training set is defined as:

$$\underline{X}_{training} = \{t_{i,1}, t_{i,2}, \ldots, t_{i,j}, \ldots, u_{k,1}, u_{k,2}, \ldots, u_{k,j}\} \tag{10}$$

The vectors in the set $X_{training}$ are used as the input of independent variables to BLR to derive the model.

$$\underline{T}_{evaluation} = \{t_{i,1}, t_{i,2}, \ldots, t_{i,j}\} \tag{11}$$

$$\underline{U}_{evaluation} = \{u_{k,1}, u_{k,2}, \ldots, u_{k,j}\} \tag{12}$$

We also created two evaluation sets: $T_{evaluation}$ and $U_{evaluation}$, using 20% of the $B$ and $V$ sets, respectively. The evaluation metric $X_{evaluation}$ is defined as:

$$\underline{X}_{evaluation} = \{t_{i,1}, t_{i,2}, \ldots, t_{i,j}, \ldots, u_{k,1}, u_{k,i}, \ldots, u_{k,j}\} \tag{13}$$

We constructed a BLR model for each attack using the monitoring parameters taken at the network layer. To derive the detection model of each attack, only the respective malicious activity was used. Table 3 shows the number of vectors used for each training and evaluation set (seen as $j$ in (10) and (13)). The vectors for the training and evaluation sets were randomly selected.

### B. EVALUATION STAGE

The objective is to capture the model by using a small set of significant variables and to achieve a high overall correct prediction percentage.

The BLR detection model for each attack was evaluated using the respective $\underline{X}_{evaluation}$ metric. The SF-FR BLR model

**TABLE 4.** BLR evaluation: Selective forward attacks.

| Evaluation Set | | True diagnosis: Viral | True diagnosis: Benign |
|---|---|---|---|
| SF-FR | Viral | 15 | 5 |
| | Benign | 0 | 20 |

Accuracy ratio (ACC) = 0.88
Recall / TPR = 0.75
Precision / PPV = 1

| Evaluation Set | | True diagnosis: Viral | True diagnosis: Benign |
|---|---|---|---|
| SF-BN | Viral | 15 | 5 |
| | Benign | 0 | 20 |

Accuracy ratio (ACC) = 0.88
Recall / TPR = 0.75
Precision / PPV = 1

was evaluated using the $\underline{X}_{evaluation}$ metric that included malicious node activity created by nodes under the influence of a SF-FR attack. An activity is considered to be benign when the probability (as shown in (4)) is less than 0.5, otherwise, the node activity is classified as viral.

#### 1) SELECTIVE FORWARD

In the training stage we created two Selective Forward BLR models, a model for Selective Forward - Forwarding Ratio (SF-FR) and a BLR model for the Selective Forward - Block Node (SF-BN), using only four independent variables. The independent variables that were used are the *number of packets forwarded* ($P_{FR}$), *data packets received* ($P_{RC}$), the *number of announcements received* ($P_{An}$), and the *number of packets dropped* ($P_{DR}$). The resulting probability model is shown in (14).

$$P = \frac{e^{\alpha+\beta_{fr}*P_{FR}i+\beta_{rc}*P_{RC}+\beta_{dr}*P_{DR}}}{1+e^{\alpha+\beta_{fr}*P_{FR}i+\beta_{rc}*P_{RC}+\beta_{dr}*P_{DR}}} \quad (14)$$

The evaluation results for both Forwarding Ratio and Block Node attacks are shown in Table 4. An alarm is raised when the result of the BLR model is more than 0.5. As expected, there are false negatives, since there are nodes within the network that are never chosen to be relay nodes. More specifically, four nodes within the network are placed the furthest away from the Sink, in this case four hops away, and they are never chosen to forward any packets. Therefore, even though they are compromised, they do not have any impact on the compromised node activity. The Precision (PPV) metric is 100% indicating that all alarms raised are classified correctly as viral activity. The Recall (TPR) shows that the model was able to detect the viral node in 75% of the malicious scenarios. The Accuracy rate is 88%, as it also takes into consideration correct classification of the benign activity. The BLR model was able to classify correctly 100% of the benign activity.

#### 2) SELECTIVE FORWARD AND BLACKHOLE

We followed the same principle of training and evaluation for analyzing the Selective Forward and Blackhole attacks. We derived two BLR models, one for each attack, and evaluated them with their corresponding evaluation sets. The results are shown in Table 5. The node activities were correctly classified to either benign or malicious. The BLR models have successfully detected all attacks having a 100% Accuracy level, 100% Recall, and 100% Precision.

**TABLE 5.** BLR evaluation: Selective forward and blackhole attacks.

| Evaluation Set | | True diagnosis: Viral | True diagnosis: Benign |
|---|---|---|---|
| SFBH-FR | Viral | 20 | 0 |
| | Benign | 0 | 20 |

Accuracy ratio (ACC) = 1
Recall / TPR = 1
Precision / PPV = 1

| Evaluation Set | | True diagnosis: Viral | True diagnosis: Benign |
|---|---|---|---|
| SFBH-BN | Viral | 20 | 0 |
| | Benign | 0 | 20 |

Accuracy ratio (ACC) = 1
Recall / TPR = 1
Precision / PPV = 1

**TABLE 6.** BLR evaluation: Sinkhole attack.

| Evaluation Set | | True diagnosis: Viral | True diagnosis: Benign |
|---|---|---|---|
| Sinkhole | Viral | 20 | 0 |
| | Benign | 0 | 20 |

Accuracy ratio (ACC) = 1
Recall / TPR = 1
Precision / PPV = 1

#### 3) SINKHOLE

At the training stage of the Sinkhole attack, the most significant parameters differ from the Selective Forward and Blackhole attacks. More precisely, there were again four significant parameters: *packets forwarded* ($P_{FR}$), *data packets received* ($P_{RC}$), *data packets sent* ($P_{PS}$), and *number of announcements received* ($P_{An}$) (see Table 1).

The Sinkhole BLR model also achieved a 100% detection Accuracy, 100% Precision and 100% Recall. The model detected and raised an alarm for all malicious activity and correctly classified all benign activity with no false alarms.

### C. COMPLEXITY

Local detection methods require monitoring local node activity and computing at real time a regression model. RMT is used as a background process that stores node activity and presents them at predefined time intervals. The computation and memory overhead of RMT are not deterrent factors for implementing the BLR regression model [30]. BLR detection models have been successfully implemented and launched in sensor nodes in [34]. Along with RMT, the regression models (for the three attacks), impose only an extra 9.58 KB of ROM and a 0.58% increase in power and energy.

## V. GLOBAL IDS USING SUPPORT VECTOR MACHINES

The current section describes the method used in the Global/Centralized SA case (see Fig. 1b).

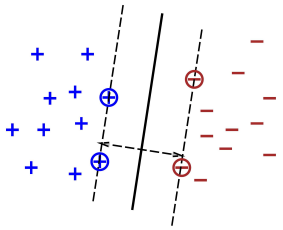

**FIGURE 3.** The optimal hyperplane that separates the positive and negative values. The position of the hyperplane is determined by the training set pairs that are closest called the support vectors [35].

SVMs maximize the margin around the decision surface, also called hyperplane. In order to do so, support vectors, also known as training sets, are used as inputs. As with any machine learning technique, there are input and output training samples. The input training sample features are denoted as $x_1$,$x_2$,...$x_l$, and the predicted value is denoted as $y_i$ for each $x_i$. The training set is labeled in pairs $(x_i, y_i), i = 1, \ldots, l$ where $x \in \mathbb{R}^n$ and $y \in \{1, -1\}$. The output is a set of weights $w_i$ one of which corresponds to each input feature and in a linear combination they predict the value of $y$.

The difference of the SVM from other neural networks is that it uses the optimization of maximizing the margin to reduce the number of weights that are nonzero to just a few that correspond to the important features and provide useful information in deciding the hyperplane (see Fig. 3). The nonzero weights correspond to the support vectors.

### A. C-SVM METHODOLOGY

The following section presents the steps taken to create the SVM detection model. It explains the reason C-Class SVM was used, the scaling performed on the input data, the kernel type, and how the free parameters were computed.

#### 1) C-SUPPORT VECTOR CLASSIFICATION

The work in [8] concluded that taking into consideration the impact of the attacks can provide insights on the network activity. Therefore, we use the C-support vector classification optimization for SVM. The C-support optimization type of SVM is used when there are more than one class of data. In our case, we have two classes: benign and malicious.

Assuming the training set $x_i \in \mathbb{R}^n$, where $i = 1, \ldots, l$; $x_i$ is separated in two classes; $y \in \mathbb{R}^l$ such that $y_i \in \{1, -1\}$; $l$ is the number of local node activity. C-SVM solves the optimization problem shown in (15) [36].

$$\begin{aligned} &\min_{\omega, b, \xi} \frac{1}{2}\boldsymbol{\omega}^\intercal \boldsymbol{\omega} + C \sum_{i=1}^{l} \xi_i \\ &\text{subject to } y_i(\boldsymbol{\omega}^\intercal \phi(\boldsymbol{x_i}) + b) \geq 1 - \xi_i \\ &\qquad \xi_i \geq 0, \quad i = 1, \ldots, l \end{aligned} \tag{15}$$

In (15), $\phi(x_i)$ maps $x_i$, into a higher-dimensional space and $C > 0$ is the regularization parameter. Due to the possible high dimensionality of the weights $w$, we solve the problem shown in (16) [36].

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^\intercal Q \boldsymbol{\alpha} - \boldsymbol{e}^\intercal \alpha \\ &\text{subject to } \boldsymbol{y}^\intercal \alpha = 0 \\ &\qquad 0 \geq \alpha_i \geq C, \quad i = 1, \ldots, l \end{aligned} \tag{16}$$

In (16), $e = [1, \ldots, 1]^\intercal$ is the vector of all ones, $Q$ is an $l$ by $l$ positive semi-definite matrix, $Q_{i,j} \equiv y_i y_j K(\boldsymbol{x_i}, \boldsymbol{x_j})$, is the kernel function. The kernel function we used for our purposes is discussed in Section V-A.3. After we solve (16), we solve equation 17 to find the optimal $w$.

$$\boldsymbol{\omega} = \sum_{i=1}^{l} y_i \alpha_i \phi(\boldsymbol{x_i}) \tag{17}$$

The decision function for the C-class SVM is shown in (18) [36].

$$sgn(\boldsymbol{\omega}^\intercal \phi(\boldsymbol{x}) + b) = sgn\left( \sum_{i=1}^{l} y_i \alpha_i K(\boldsymbol{x_i}, \boldsymbol{x}) + b \right) \tag{18}$$

#### 2) SCALING

Scaling is applied to both training and evaluation data sets to construct the data in a form that the SVM can take as input and construct the hyperplane. We use scaling such as to avoid certain parameters that are in greater numeric ranges, dominate those in smaller numeric ranges [36]. At the same time, applying scaling returns the parameters that are at most important to retrieve better results [36].

We apply scaling for all support vector $(x_i, y_i)$ using the following:

$$\boldsymbol{x_i'} = (\beta - \alpha) \frac{x_i - \min x}{\max x - \min x} + a \tag{19}$$

where $\boldsymbol{x_i'}$ is the scaling of parameter $i > 0$ and $x > 0$. Also $\min x$ and $\max x$ is the minimum and maximum numbers of the input vector. The constants $[\alpha, \beta]$ correspond to the scaling range in our case $[-1, 1]$. The scaling did not change the labels $y_i$.

Scaling our data returns the support vectors with the significant parameters that are later used as input to our SVM. Parameters that were scaled to 0 are not included as they do not provide any useful information to our SVM model.

#### 3) RADIAL BASIS FUNCTION KERNEL

There are various kernel functions one can use for SVM. We chose the Radial Basis Function (RBF) kernel, as it is suitable for non-linear data and label classification. Our selection was also influenced by initial experimentation in which the RBF kernel gave us better results than the linear kernel function and exhibited less complexity than other kernel functions, such as the sigmoid and polynomial kernel. The resulting RBF kernel function we used is shown in (20).

$$\begin{aligned} &K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \exp(-\gamma \parallel \boldsymbol{x_i} - \boldsymbol{x_j} \parallel^2), \\ &\gamma > 0, \quad \gamma = \frac{1}{2\sigma^2} \\ &\sigma \text{ is a free parameter} \end{aligned} \tag{20}$$

#### 4) CROSS-VALIDATION AND GRID SEARCH

The parameters $C$ and $\gamma$ used in (16) and (20), are free parameters for the SVM and RBF kernel [36]. To find the values of $(C, \gamma)$ an experimental search based on our data is conducted. The values of $(C, \gamma)$ should be chosen so as to avoid over fitting of our data. Over-fitting our data means mapping the data closer to the threshold line thus minimizing the margin from our hyperplane, instead of maximizing it, which is the goal of SVM [36].

To avoid over-fitting by selecting more appropriate values for $(C, \gamma)$ we use the cross-validation and grid search method. The cross validation technique separates the input data to training and evaluation. With the grid search, various pairs of $(C, \gamma)$ are evaluated to find the best pair. The method of cross-validation and grid search is repeated until the best pair of $(C, \gamma)$ is found.

### B. TRAINING AND EVALUATION

To construct our SVM model we used the Matlab LIBSVM tool [36] and local node data taken from RMT.

#### 1) TRAINING

In our case we have two classes, the benign and the viral class, denoted $-1$ and 1 respectively.

To construct the training model we use the following steps:

1) Label data into benign and malicious in the form of vectors $(x_i, y_i)$
2) Apply scaling to the training set
3) Apply RBF kernel to the training set
4) Apply cross validation and grid search to the training set
5) Input the training set and $(C, \gamma)$ values to create the SVM model

Based on the attack we use for our training model, the significant parameters varied. For Selective Forward attacks, the scaling step returns data for the parameters *data packets received* ($P_{RC}$), *packets forwarded* ($P_{FR}$), *number of packets dropped* ($P_{DR}$) and the *number of announcements received* ($P_{An}$). For the Sinkhole attack, the scaling step returns the training set for all five parameters shown in Table 1. The results of the free parameters $(C, \gamma)$ are shown in the column titled Significant Parameters in Table 7.

#### 2) EVALUATION OF SVM TRAINING MODEL

At the training stage we created an SVM model with the adjacent $(C, \gamma)$ to be used for the evaluation stage. The data sets that were used for the evaluation are shown in Table 3.

To set up our data evaluation sets we applied the following steps:

1) Label data into benign and malicious in the form of vectors $(x_i, y_i)$
2) Apply scaling to the evaluation set

**TABLE 7. SVM $(C, \gamma)$ values from our training results.**

| | Significant Parameters | |
|---|---|---|
| **Training Set** | $C$ | $\gamma$ |
| Selective Forward - Forwarding Ratio | 128.0 | 0.5 |
| Selective Forward - Block Node | 0.03125 | 0.5 |
| Selective Forward + Blackhole - Forwarding Ratio | 0.03125 | 0.0078 |
| Selective Forward + Blackhole - Block Node | 2.0 | 0.0078 |
| Sinkhole | 8.0 | 2.0 |

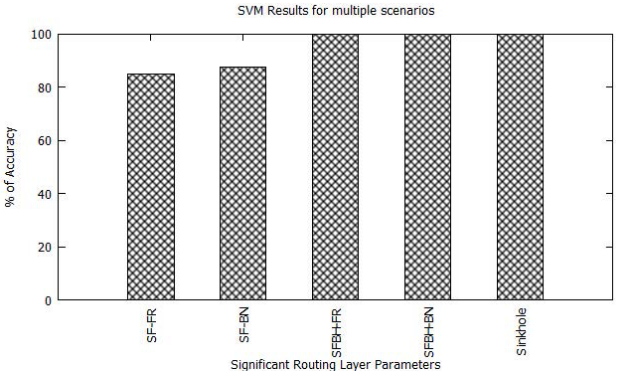


**FIGURE 4. Percentage of accuracy of SVM model per malicious scenario when using only significant parameters.**

**TABLE 8. SVM evaluation: Selective forward - forwarding ratio.**

| Evaluation Set | | True diagnosis: Viral | True diagnosis: Benign |
|---|---|---|---|
| SF-FR | Viral | 19 | 1 |
| | Benign | 5 | 19 |

Accuracy ratio (ACC) = 0.85
Recall / TPR = 0.95
Precision / PPV = 0.79

**TABLE 9. SVM evaluation: Selective forward - block node.**

| Evaluation Set | | True diagnosis: Viral | True diagnosis: Benign |
|---|---|---|---|
| SF-BN | Viral | 20 | 0 |
| | Benign | 5 | 15 |

Accuracy ratio (ACC) = 0.86
Recall / TPR = 1
Precision / PPV = 0.8

3) Input the evaluation sets and $(C, \gamma)$ values that we retrieve from the training stage to the SVM model

The results of the SVM model are shown in Fig. 4. The SF notation stands for Selective Forward, FR is Forwarding Ratio, BN is Block Node, SF-BH is Selective Forward and Blackhole attack.

In Selective Forward malicious activity the SVM did not entirely correctly classify node activity as in the Selective Forward with Blackhole and Sinkhole attacks. The Selective Forward attack drops packets that should have otherwise been forwarded. The SVM model has a Precision of 95% in SF-FR and 100% in SF-BN with the cost of creating false positives. As a result, Recall is only 80% and 79%, respectively.

**TABLE 10.** SVM evaluation: Selective forward and blackhole - forwarding ratio.

| Evaluation Set | | True diagnosis: Viral | True diagnosis: Benign |
|---|---|---|---|
| SFBH-FN | Viral | 20 | 0 |
| | Benign | 0 | 20 |

Accuracy ratio (ACC) = 1
Recall / TPR = 1
Precision / PPV = 1

**TABLE 11.** SVM evaluation: Selective forward and blackhole - block node.

| Evaluation Set | | True diagnosis: Viral | True diagnosis: Benign |
|---|---|---|---|
| SFBH-BN | Viral | 20 | 0 |
| | Benign | 0 | 20 |

Accuracy ratio (ACC) = 1
Recall / TPR = 1
Precision / PPV = 1

**TABLE 12.** SVM evaluation: Sinkhole.

| Evaluation Set | | True diagnosis: Viral | True diagnosis: Benign |
|---|---|---|---|
| Sinkhole | Viral | 20 | 0 |
| | Benign | 0 | 20 |

Accuracy ratio (ACC) = 1
Recall / TPR = 1
Precision / PPV = 1

In the Selective Forward with Blackhole and the Sinkhole attacks, the results show that the SVM models have correctly classified all activities (see 10, 11, and 12).

### C. COMPLEXITY

SVM models have proven to be good indicators of whether node activity is malicious or benign, but they come at the cost of high complexity. According to [35], the SVM computation overhead depends on the number of support vectors. In the best case scenario, in which we know beforehand the support vectors, to determine the coefficients of the support vectors by a system of $R$ linear functions we require a number of operations proportional to $R^3$. The cost of complexity increases if the support vectors need to be discovered. The authors in [35] also state that the kernel function has a high computation cost as well.

SVM computational complexity overhead makes it difficult, if not impossible, to be implemented in a constrained node. The authors [10] and [17] apply SVM machines in non-constrained nodes to detect possible intervention to the network or data outliers.

Placing SVM at a global position, comes with the cost of creating extra network traffic. The central node/gateway can detect malicious activity only based on information it has locally or has received. Network nodes should make available to the gateway their activity, thus imposing a communication overhead in the detection process.

## VI. DECENTRALIZED IDS USING THRESHOLDS

The current section presents a decentralized security agent approach, as shown in Fig. 1c. We implemented the work proposed by [9] using monitoring nodes to detect malicious intervention within the network. More precisely, we applied the retransmission rule that was designed to detect Selective Forward and Blackhole attacks [9]. The sinkhole attack cannot be recognized using this method. The monitoring node eavesdrops network traffic and tracks a packet's status. When a packet is sent to a neighboring node to be forwarded, the monitoring node checks whether the relay node has successfully forwarded the packet as intended. If the relay node does not forward the message, or if a threshold of packets not forwarded is exceeded, then an alarm is raised.

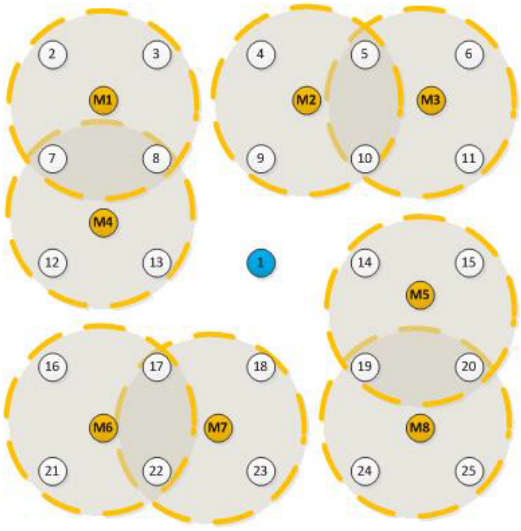


**FIGURE 5.** Placement of security agents using thresholds.

**TABLE 13.** Monitoring blocks.

| Monitoring Nodes | Neighboring Node ID | Threshold: Max | Threshold: Avg | Threshold: Min |
|---|---|---|---|---|
| M1 | 2, 3, 7, 8 | 23 | 5 | 0 |
| M2 | 4, 5, 9, 10 | 175 | 98 | 15 |
| M3 | 5, 6, 10, 11 | 66 | 28 | 5 |
| M4 | 7, 8, 12, 13 | 187 | 82 | 10 |
| M5 | 14, 15, 19, 20 | 165 | 61 | 11 |
| M6 | 16, 17, 21, 22 | 51 | 13 | 1 |
| M7 | 17, 18, 22, 23 | 142 | 77 | 5 |
| M8 | 19, 20, 24, 25 | 90 | 18 | 1 |

The position of the monitoring nodes should be such that each regular node is covered by at least one monitoring node. Eight monitoring nodes were placed in key positions in the network topology to achieve full coverage, as shown in Fig. 5. In certain cases there is overlapping between the monitoring nodes' range, and some nodes are monitored by more than one node. The nodes that each monitoring node is responsible for are listed in Table 13. This is a very tedious task.

Each monitoring node is responsible for listening to the traffic between its neighboring nodes, saving the packets sent and packets forwarded by each node. At predefined time intervals the monitoring node evaluates the number of packets sent and the number of packets forwarded from and to a specific node. To evaluate the exact number of packets sent versus the number of packets forwarded, the monitoring node

**Algorithm 1** Monitoring Node: Detection of Non-Forwarded Messages

```
for Each neighboring node i do
    Listen to traffic by node i
    if Sent Packets then
        Update table packets sent
    end if
    if Forwarded Packets then
        Update table packets forwarded
    end if
end for
if Detection Time Interval is expired then
    for Each neighboring node i do
        for Each entry in the packets sent table do
            if Next hop node j is a neighboring node then
                for Each entry in the packets forwarded of
  j node do
                    if Packet not found then
                        Update packets not found variable
                    else
                        break
                    end if
                end for
            end if
        end for
        if Packets Not found is more than threshold then
            Raise Alarm!
        end if
    end for
end if
```

needs to have within its range both the sender node and the relay node. An alarm is raised when the number of packets not forwarded exceeds a predefined threshold. The algorithm we followed for the monitoring node is shown in Algorithm 1.

### A. THRESHOLDS

As proposed by the work in [9], the thresholds are established using benign information only. It is assumed that at the beginning of the simulation the network is benign and for a short period of time the monitoring nodes listen to the traffic and construct their thresholds. For our work, we simulated ten times a network in which the Sink is in the middle using random seeds and using only the benign applications we define the required thresholds. To construct the thresholds for each monitoring node, we averaged the results of packets not forwarded. For the purpose of our evaluation we derived three thresholds for each monitoring node; the minimum, average and maximum number of packets not forwarded (shown in Table 13).

### B. EVALUATION

Similarly to the previous cases, for the evaluation stage we conducted simulations of 15 minutes, out of which the first two minutes were considered as the initialization stage and no monitoring was taking place. At the initialization stage the network is created using the WSP routing protocol. Each minute the monitoring node compares the packets forwarded and sent for each of the neighboring nodes following Algorithm 1.

We conducted a total of 96 malicious experiments for the topology in which the Sink is in the middle. The malicious scenarios included one malicious node that would be infected either with the Selective Forward attack or with the Selective Forward and Blackhole attack. We again used the two variations of the Selective Forward attacks; the Block Node attack and the Forwarding Ratio attack.

The results presented correspond to the case where the thresholds in the monitoring node were set to the maximum number of packets forwarded shown in Table 13. The max threshold created the least number of false alarms.

#### 1) SELECTIVE FORWARD

The confusion matrices in Table 14 show the results for the Selective Forward attacks. The Accuracy rates of both Forwarding Ratio (FR) and Block Node (BN) are 90% as the monitoring nodes were able to correctly classify most of the activity to the correct type. However, the Recall rate was 13% for the FR and 17% for the BN, indicating that there

**TABLE 14.** Decentralized evaluation: Selective forward.

| Evaluation Set | | True diagnosis: Viral | True diagnosis: Benign |
|---|---|---|---|
| SF-FR | Viral | 3 | 21 |
| | Benign | 39 | 513 |

Accuracy ratio (ACC) = 0.9
Recall / TPR = 0.13
Precision / PPV = 0.07

| Evaluation Set | | True diagnosis: Viral | True diagnosis: Benign |
|---|---|---|---|
| SF-BN | Viral | 4 | 20 |
| | Benign | 40 | 512 |

Accuracy ratio (ACC) = 0.9
Recall / TPR = 0.17
Precision / PPV = 0.09

**TABLE 15.** Decentralized evaluation: Selective forward and blackhole attacks.

| Evaluation Set | | True diagnosis: Viral | True diagnosis: Benign |
|---|---|---|---|
| SFBH-FR | Viral | 17 | 7 |
| | Benign | 33 | 519 |

Accuracy ratio (ACC) = 0.93
Recall / TPR = 0.71
Precision / PPV = 0.34

| Evaluation Set | | True diagnosis: Viral | True diagnosis: Benign |
|---|---|---|---|
| SFBH-BN | Viral | 17 | 7 |
| | Benign | 30 | 522 |

Accuracy ratio (ACC) = 0.94
Recall / TPR = 0.71
Precision / PPV = 0.362

**TABLE 16.** Detection models.

| Approach | Accuracy Rate | | | Recall / TPR | | | Precision / PPV | | |
|---|---|---|---|---|---|---|---|---|---|
| | SF | SF & BH | Sinkhole | SF | SF & BH | Sinkhole | SF | SF & BH | Sinkhole |
| Local Detection | 87.5% | 100% | 100% | 75% | 100% | 100% | 100% | 100% | 100% |
| Global Detection | 86% | 100% | 100% | 100% | 100% | 100% | 80% | 100% | 100% |
| Decentralised Detection | 90% | 94% | N/A | 17% | 71% | N/A | 9% | 36% | N/A |

was a high ratio of misclassifications. The FR Precision is 7% and 9% for the BN as most of the malicious activity were undetected.

#### 2) SELECTIVE FORWARD AND BLACKHOLE

The Selective Forward and Blackhole attack returned higher values on the evaluations metrics compared with the Selective Forward attacks. The Accuracy rates for FR and BN were 93% and 94% respectively, the Recall values were 71% for both attacks and the Precision values were 34% and 36.2% (shown in Table 15).

### C. COMPLEXITY

The framework of using thresholds to detect the presence of malicious attack is promising as it has achieved approximately more than 90% Accuracy levels and in certain cases the monitoring nodes were able to detect which node is malicious within the network. However, the Recall and Precision rates were low, showing that a lot of false alarms were created. To apply the threshold algorithm requires more dedicated nodes within the network, which need to be placed at key locations. The thresholds of the monitoring nodes need to be different and depend on the location of the monitoring node (see Table 13).

The threshold algorithm requires additional memory and has computational overhead. In the worst case scenario the computational cost of Algorithm 1 can reach up to $O(N^2)$. The monitoring nodes require memory to store the number of packets sent and packets forwarded for each neighboring node.

## VII. CONCLUSION

In this work we have evaluated three anomaly-based IDS for IoT, each one placing its security agent(s) at different location(s) within the network. For the local SA method we used a technique based on the Binary Logistic Regression. BLR was proven suitable to be installed at a node, to monitor, and to detect abnormalities with negligible computational cost and without any communication cost. SMV was considered in the context of global SA method and a threshold-based technique was used as an example of a distributed SA solution. Table 16 shows a synopsis of the highest Accuracy achieved by each technique. BLR provides the opportunity to have only one parameter to set and offers both high detection rates (accuracy, recall and precision) and easy implementation [34]. Threshold-based detection is easier to set, but needs a lot of trial and error to find the correct levels for the monitored parameters and is quite sensitive to topology and application changes. It also requires dedicated nodes to be placed at key positions and to listen promiscuously to neighbor traffic (see Fig. 1c). c-SVM, on the other hand, does provide high accuracy levels, but it is complex to setup and does not offer any advantage over BLR. SVM has a higher communication cost compared to BLR and Threshold, as its nodes have to communicate their local activity to the global security agent (see Fig. 1b). On the contrary, BLR is installed within the node and monitoring and detection are done automatically without having to report to a central node or extra hardware (see Fig. 1a).

## REFERENCES

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.
[2] "Baseline security recommendations for IoT in the context of critical information infrastructures," Eur. Union Agency Netw. Inf. Secur., Heraklion, Greece, 2017.
[3] H. Cavusoglu, B. Mishra, and S. Raghunathan, "A model for evaluating IT security investments," *Commun. ACM*, vol. 47, no. 7, pp. 87–92, Jul. 2004.
[4] T. Stibor, P. Mohr, J. Timmis, and C. Eckert, "Is negative selection appropriate for anomaly detection?" in *Proc. 7th Annu. Conf. Genetic Evol. Comput.*, Jun. 2005, pp. 321–328.
[5] E. Benkhelifa, T. Welsh, and W. Hamouda, "A critical review of practices and challenges in intrusion detection systems for IoT: Toward universal and resilient systems," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3496–3509, Feb. 2018.
[6] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *J. Netw. Comput. Appl.*, vol. 84, pp. 25–37, Apr. 2017.
[7] I. Butun, S. D. Morgera, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 266–282, 1st Quart., 2014.
[8] C. Ioannou and V. Vassiliou, "The impact of network layer attacks in wireless sensor networks," in *Proc. Int. Workshop Secure Internet Things*, Crete, Greece, Sep. 2016, pp. 20–28.
[9] A. P. R. da Silva, M. H. T. Martins, B. P. S. Rocha, A. A. F. Loureiro, L. B. Ruiz, and H. C. Wong, "Decentralized intrusion detection in wireless sensor networks," in *Proc. 1st ACM Int. Workshop Qual. Service Secur. Wireless Mobile Netw.*, Oct. 2005, pp. 16–23.
[10] S. Kaplantzis, A. Shilton, N. Mani, and Y. A. Sekercioglu, "Detecting selective forwarding attacks in wireless sensor networks using support vector machines," in *Proc. 3rd Int. Conf. Intell. Sensors, Netw. Inf. (ISSNIP)*, Dec. 2007, pp. 335–340.
[11] S. Schaust and H. Szczerbicka, "Misbehaviour detection for wireless sensor networks-necessary or Not?" *Fachgespräch Sensornetzwerke*, vol. 1, p. 51, Aug. 2007.
[12] C. Ioannou, V. Vassiliou, and C. Sergiou, "An intrusion detection system for wireless sensor networks," in *Proc. 24rd Int. Conf. Telecommun. (ICT)*, May 2017, pp. 1–5.
[13] Y. Huang, L. Zhang, G. Lian, R. Zhan, R. Xu, Y. Huang, B. Mitra, J. Wu, and G. Luo, "A novel mathematical model to predict prognosis of burnt patients based on logistic regression and support vector machine," *Burns*, vol. 42, no. 2, pp. 291–299, Mar. 2016.
[14] F. Huang, Z. Jiang, S. Zhang, and S. Gao, "Reliability evaluation of wireless sensor networks using logistic regression," in *Proc. Int. Conf. Commun. Mobile Comput. (CMC)*, vol. 3, Apr. 2010, pp. 334–338.
[15] O. Salem, A. Guerassimov, A. Mehaoua, A. Marcus, and B. Furht, "Sensor fault and patient anomaly detection and classification in medical wireless sensor networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2013, pp. 4373–4378.

[16] C. Ioannou, "The hunt for viral processes," M.S. thesis, Dept. Comput. Sci., Florida Inst. Technol., Melbourne, FL, USA, 2003.
[17] Y. Zhang, N. Meratnia, and P. Havinga, "Adaptive and online one-class support vector machine-based outlier detection techniques for wireless sensor networks," in *Proc. Int. Conf. Adv. Inf. Netw. Appl. Workshops*, May 2009, pp. 990–995.
[18] R. Muraleedharan and L. Osadciw, "Security: Cross layer protocol in wireless sensor network," in *Proc. 25th IEEE Int. Conf. Comput. Commun.*, Apr. 2006, pp. 1–2.
[19] F. Liu, X. Cheng, and D. Chen, "Insider attacker detection in wireless sensor networks," in *Proc. 26th IEEE Int. Conf. Comput. Commun.*, May 2007, pp. 1937–1945.
[20] I. Onat and A. Miri, "An intrusion detection system for wireless sensor networks," in *Proc. IEEE Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, vol. 3. Aug. 2005, pp. 253–259.
[21] N. Aschenbruck, J. Bauer, J. Bieling, A. Bothe, and M. Schwamborn, "A security architecture and modular intrusion detection system for WSNs," in *Proc. 9th Int. Conf. Networked Sens. (INSS)*, Jun. 2012, pp. 1–8.
[22] N. Alrajei, H. Fu, and Y. Zhu, "Information Theory Based Intrusion Detection in Wireless Sensor Networks," *J. Commun. Technol., Electron. Comput. Sci.*, vol. 5, pp. 11–21, Apr. 2016.
[23] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki—A lightweight and flexible operating system for tiny networked sensors," in *Proc. 29th Annu. IEEE Int. Conf. Local Comput. Netw.*, Nov. 2004, pp. 455–462.
[24] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with COOJA," in *Proc. 1st IEEE Int. Workshop Practical Issues Building Sensor Netw. Appl.*, Nov. 2006, pp. 641–648.
[25] S. Raza, L. Wallgren, and T. Voigt, "SVELTE: Real-time intrusion detection in the Internet of Things," *Ad Hoc Netw.*, vol. 11, no. 8, pp. 2661–2674, May 2013. [Online]. Available: http://dx.doi.org/10.1016/j.adhoc.2013.04.014
[26] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015.
[27] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," in *Proc. 1st IEEE Int. Workshop Sensor Netw. Protocols Appl.*, May 2002, pp. 113–127.
[28] I. Krontiris, T. Dimitriou, T. Giannetsos, and M. Mpasoukos, "Intrusion detection of sinkhole attacks in wireless sensor networks," in *Algorithmic Aspects Wireless Sensor Network*. Berlin, Germany: Springer, 2008.
[29] S. Y. Moon and T. H. Cho, "Intrusion detection scheme against sinkhole attacks in directed diffusion based sensor networks," *Int. J. Comput. Sci. Netw. Secur.*, vol. 9, no. 3, pp. 118–122, 2009.
[30] C. Ioannou, V. Vassiliou, and C. Sergiou, "RMT: A wireless sensor network monitoring tool," in *Proc. 13th ACM Symp. Perform. Eval. Wireless Ad Hoc, Sensor, Ubiquitous Netw.*, Nov. 2016, pp. 45–49.
[31] A. Azmoodeh, A. Dehghantanha, M. Conti, and K.-K. R. Choo, "Detecting crypto-ransomware in IoT networks based on energy consumption footprint," *J. Ambient Intell. Humanized Comput.*, vol. 9, no. 4, pp. 1141–1152, Aug. 2017.
[32] E. Kabir, J. Hu, H. Wang, and G. Zhuo, "A novel statistical technique for intrusion detection systems," *Future Gener. Comput. Syst.*, vol. 79, no. 1, pp. 303–318, 2018.
[33] C. Cervantes, D. Poplade, M. Nogueira, and A. Santos, "Detection of sinkhole attacks for supporting secure routing on 6LoWPAN for Internet of Things," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2015, pp. 606–611.
[34] C. Ioannou and V. Vassiliou, "An intrusion detection system for constrained WSN and IoT nodes based on binary logistic regression," in *Proc. 21st ACM Int. Conf. Model., Anal. Simul. Wireless Mobile Syst.*, Nov. 2018, pp. 259–263.
[35] L. Bottou and C.-J. Lin, "Support vector machine solvers," in *Large Scale Kernel Machines*, L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, Eds. Cambridge, MA, USA: MIT Press, 2007, pp. 301–320.
[36] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, 2011.

**CHRISTIANA IOANNOU** received the B.Sc. degree in computer science from San Diego State University, the M.Sc. degree in computer science from the Florida Institute of Technology, the Ph.D. degree from the Department of Computer Science, University of Cyprus, in 2017, and the postgraduate diploma degree in management from the Mediterranean Institute of Management. She completed the master's degree through the CASP scholarship from the Fulbright Commission in Cyprus. Her master's thesis topic was "The Hunt for Viral Processes". Her Ph.D. thesis title is "mIDS: an Intrusion Detection System for Wireless Sensor Networks and the Internet of Things".

She is currently a Postdoctoral Researcher with the Network Research Laboratory (NetRL), Department of Computer Science, and at the Research Centre on Interactive Media Smart Systems and Emerging Technologies (RISE), University of Cyprus. She is also a part time Lecturer with the Computer Science Department and the Multimedia Department, University of Nicosia. She is the course Leader for Network Security Defenses and Countermeasures master course and also a Teacher in Ethical Hacking. Her research interests include WSNs and the IoT focusing on Security, IDSs using anomaly and signature-based detection mechanisms for novel and known attacks.

**VASOS VASSILIOU** received the undergraduate degree in electrical engineering from the Higher Technical Institute (HND), in 1993, and the B.Sc. degree in electrical engineering from the University of South Florida, in 1997, and the M.Sc. and Ph.D. degrees in electrical and computer engineering from the Georgia Institute of Technology, in 1999 and 2002, respectively. He did his undergraduate studies with the CASP scholarship from the Cyprus Fulbright Commission, Georgia Institute of Technology.

He is currently a Faculty Member and the Co-Director of the Networks Research Laboratory, Computer Science Department, University of Cyprus. He is also the Group Leader of the Smart Networked Systems Research Group of the newly formed RISE Center of Excellence on interactive Media, Smart Systems and Emerging technologies, Nicosia, Cyprus. He has well-published on topics that include next-generation network architectures, mobile protocols, mobile networks, wireless communications and QoS, and traffic engineering for computer and telecommunication networks. His current research interests include next-generation networks and the Internet of Things domain, where he and his team work on security, mobility, traffic congestion and flow-control, and data management issues.

Dr. Vassiliou is a Senior Member of the IEEE and a member of the ACM and participates in the technical program committees of several international conferences, such as GLOBECOM, PIMRC, VTC, WCNC, and others. He is the Chair of the IEEE ComSoc Cyprus Chapter, part of the IBM Academic Initiative, and the Academic Advocate for the ISACA Cyprus Chapter.