Classifying Security Attacks in IoT Networks Using Supervised Learning

Christiana Ioannou and Vasos Vassiliou
Department of Computer Science, University of Cyprus and
RISE - Research Center on Interactive Media, Smart Systems and Emerging Technologies
Nicosia, Cyprus
{cioannou,vasosv}@cs.ucy.ac.cy

***Abstract*—Machine learning models have long be proposed to detect the presence of unauthorized activity within computer networks. They are used as anomaly detection techniques to detect abnormal behaviors within the network. We propose to use Support Vector Machine (SVM) learning anomaly detection model to detect abnormalities within the Internet of Things. SVM creates its normal profile hyperplane based on both benign and malicious local sensor activity. An important aspect of our work is the use of actual IoT network traffic with specific network-layer attacks implemented by us. This is in contrast to other works creating supervised learning models, with generic datasets. The proposed detection model achieves up to 100% accuracy when evaluated with unknown data taken from the same network topology as it was trained and 81% accuracy when operating in an unknown topology.**

***Index Terms*—Internet of Things, IoT, Routing Attacks, Intrusion Detection Systems, Support Vector Machines.**

I. Introduction

The Internet of Things (IoT) may be the most unsecure type of network encountered nowadays. This is especially alarming if we realize that the IoT is increasingly becoming a bigger part of our everyday routine. Things are now connected to each other forming their own network in a user's private life, which the user can access through the Internet. IoT applications include smart cities, intelligent transportation, responsive environments, and many more, which are, in certain cases, beyond the users' direct control. A computer owner can theoretically control the level of security in their computer by adapting security measures, not downloading certain applications, using security software etc. A citizen in a Smart City cannot control the security measures employed in the IoT-based smart city environment and thus can not control their exposure level.

In general there exist three security layers to establish a secure network. The preventive, detective and response layer, applied in the same order. The first line of defense is the prevention layer, usually equipped with rules on what is allowed to enter the network or a device. The rules may be applied automatically through the use of a firewall, or sometimes by the user's judgment. Nevertheless, they are certain cases in which a new attack can penetrate this layer of defence and infect the host and/or the network.

This research is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°739578 and the government of the Republic of Cyprus through the Directorate General for European Programmes, Coordination and Development.

Intrusion detection systems (IDS) are the second line of defense in computer systems and networks. They recognize the attacks that have already penetrated the system's preventive measures. Detection mechanisms are generally classified as pattern detection-based and anomaly detection-based. The current work focuses on anomaly detection, which does not require knowledge of the existing attack, as in the pattern detection. An IDS raises an alarm when it detects an abnormal or never seen behavior. Once an alarm is raised, the third security layer is enabled. Automatic scripts or human intervention is require to analyse the alarm and respond to the potential threat.

Despite the maturity of IDS technology for traditional networks, current solutions are inadequate for WSN and IoT systems, because of IoT particular characteristics that affect IDS development. At first, processing and storage capacity of network nodes that host IDS agents is an important issue. In traditional networks, the system administrator deploys IDS agents in nodes with higher computing capacity. IoT networks are usually composed of nodes with resource constraints. Therefore, finding nodes with the ability to support IDS agents is harder in IoT systems. The second particular characteristic is related to the network architecture. In traditional networks, end systems are directly connected to specific nodes (e.g., wireless access points, switches, and routers) that are responsible for forwarding the packets to the destination. IoT networks, on the other hand, are usually multihop. Then, regular nodes may simultaneously forward packets and work as end systems. For instance, in IoT networks, sensors with short-range communication capabilities are deployed in a structured or random topology. Then, the data collected by a sensor is forwarded through a path of sensors until reaching a gateway to the Internet. This kind of architecture poses new challenges for IDSs. The last characteristic is related to specific network protocols. IoT networks use protocols that are not employed in traditional networks, such as IEEE 802.15.4, IPv6 over Low-power Wireless Personal Area Network (6LoWPAN), IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) and Constrained Application Protocol (CoAP). Different protocols bring original vulnerabilities and new demands for IDS.

Anomaly intrusion detection techniques can be classified based on the methods used for profiling normal behavior and capturing abnormalities. The main anomaly profiling techniques can be categorized as applying thresholds, game theory, fuzzy logic, machine learning, and biologically inspired techniques. The current work proposes c-Support Vector Machines (SVM) to be used as an IDS to detect routing layer attacks. The c-SVM model creates its detection model by training it with both benign and malicious activity. It uses local node activity from the nodes within the network and manages to have up to 100% accuracy in detecting the malicious node.

The rest of the paper is structured as follows: Section II discusses other machine learning detection techniques. Section III presents SVM and Section IV presents the training and evaluations results of the proposed solution. Section V concludes the current work.

II. Related Work

Support Vector Machine (SVM) are a class of machine learning algorithms that can create a binary classification model out of complex highly non-linear problems [1]. An SVM performs classification tasks by constructing hyperplanes in a multidimensional space that separates cases of different class labels. Many researchers have used SVM to detect possible intruders in the network such as the work in [1]–[3].

The work in [3], proposes a multi-class SVM IDS system and the work in [2] proposes a multi-class least square SVM (LS-SVM) IDS model. Both works are evaluated using traditional Internet traffic datasets (KDD-99). We believe that those types of datasets are not representative of traffic found in WSNs. However, they are suitable for showcasing the efficacy of the SVM solution. Similar to the current work these past works use activity from both benign and malicious activity to train their system. The class labels of this dataset define either a "normal" state of the system or one of four types of attacks: "probe,""denial of service (DoS),""user to root (U2R),"and "remote to local (R2L)." In the current work we are using a binary classification, thus we label the training data only as benign or malicious.

In the work of [1], they use one class SVMs as they trained their system using only the normal behavior of the network and sensing data. They support that by only using benign behavior, an unknown attack will be detected. 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 to not impose memory and power overhead in the constrained nodes.

The constrained sensor nodes in the network transmit a packet to the Sink node when they sense movement. The Sink node is responsible for analysing the movement pattern of the intruder within the network. It analyses the incoming bandwidth utilization and the number of hops each message took to reach the Sink.

The SVM detection technique of [1] uses bandwidth and hop count to detect Blackhole and Selective Forward attacks. They concluded that the Blackhole attack is detected with 100% accuracy, whereas the Selective Forward attack is detected with 85% accuracy when the intruder drops packets coming from 80% of the nodes. However, their accuracy decreases as the selective forward attack drops packets coming from 30% and 50% of the nodes.

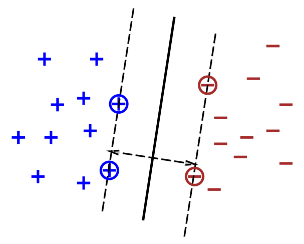


Fig. 1: The optimal hyperplane that separates the positive and negative values. The position of the hyperplane is determined by the training set pairs that are closest called the support vectors [7].

The work in [4], proposes a decentralized SVM within a Wireless Sensor Network (WSN) in which the SVM model is build and updated outside the network. The model's support vectors are transmitted to the cluster heads. The training data was classified as benign and malicious as in the current work. However, the work was evaluated using the KDD-99 database; whereas, the current work is evaluated using routing layer parameters taken from constrained sensor nodes executing either benign or malicious application.

The current work creates and evaluates the c-SVM model for the same network topologies as found in [5]. The attacks used are taken from the work of [6], namely Selective Forward, Blackhole and Sinkhole.

III. Support Vector Machines

The current section describes the methods used to create our SVM model. SVMs maximize the margin around the decision surface, also called hyperplane. In order to maximize it, support vectors, also known as training sets, are used as inputs.

As with any machine learning technique, there exist 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 machine learning techniques, is that it uses the optimization of maximising 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 Figure 1). The nonzero weights correspond to the support vectors.

A. *C-Support Vector Classification*

The work in [6] concluded that taking into consideration the impact on the attacks can provide insights on the network

TABLE I: Routing parameters used as input in the SVM

Routing parameters	
Data Packets Received	Data Packets Sent
Packets Forwarded	Packets Dropped
Announcements Received	

behavior. Therefore, we use the C-support vector classification optimisation for SVM. The C-support optimization type of SVM is used when there are more than one classes of data. In our case, we have two classes: the benign and the malicious.

The decision function for the C-class SVM is shown in equation (1) [8].

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

Where $\phi(x_i)$ maps x_i into a higher-dimensional space, $K(\boldsymbol{x_i},\boldsymbol{x_j})$, is the kernel function. The kernel function we used for our purposes is discussed in section III-C and b is a coefficient. For more information on how to derive the decision function refer to [8].

B. 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 in order to prevent certain parameters that are in greater numeric ranges to dominate those in smaller numeric ranges [8]. At the same time, applying scaling returns the parameters that are at most important to retrieve better results [8].

We apply scaling for all support vector (x_i, y_i) using the following equation (2):

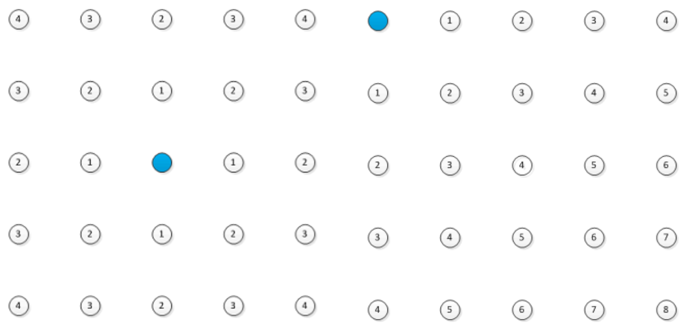
$$\boldsymbol{x_i'} = (\beta-\alpha)\frac{x_i - \min x}{\max x - \min x} + a \tag{2}$$

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 model. Parameters that were scaled to 0 are not included as they do not provide any useful information to our SVM model.

C. Radial Basis Function kernel

There are various kernel functions we can use for our SVM. We choose the Radial Basis Function (RBF) kernel to implement our SVM. RBF kernel is used for non-linear data and label classification. We chose the RBF kernel as it gave us better results than the Linear kernel function and has less complexity than other kernel functions, such as the sigmoid



(a) Sink in the middle of the grid (b) Sink on top left corner of the grid

Fig. 2: Network Topology and placement of the Sink node

and polynomial kernel. The RBF kernel function we used is shown in equation (3).

$$\begin{aligned} &K(\boldsymbol{x_i},\boldsymbol{x_j}) = \exp(-\gamma \parallel \boldsymbol{x_i} - \boldsymbol{x_j} \parallel^2), \\ &\gamma > 0, \gamma = \frac{1}{2\sigma^2} \\ &\sigma \text{ is a free parameter} \end{aligned} \tag{3}$$

D. Cross-Validation and Grid Search

The parameters C and γ used to derive the decision function and the kernel function shown in (3) are free parameters for the SVM and RBF kernel [8]. To find the values of (C, γ) an experimental search based on our data is conducted. The values of (C, γ) 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 minimising the margin from our hyperplane, instead of maximising, which is the goal of SVM [8].

To avoid over-fitting by selecting more appropriate values for (C, γ) 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, γ) are evaluated to find the best pair. The method of cross-validation and grid search is repeated until the best pair of (C, γ) is found.

IV. Results

There are two basic stages of SVM, the training stage and the evaluation state. At the training stage, the SVM takes as input training data and creates a hyperplane. The classification hyperplane model is evaluated at the evaluation stage in which the model created takes as input new data and maps them in the hyperplane. The end result of the evaluation stage is the classification of the data in the model. The parameters we used as input in both the training and evaluation stage were taken from each sensor (shown in Table I).

Two network grid topologies used are shown in Figures 2a and 2b. Figure 2a, places the Sink in the middle and is considered to be the best case scenario as a packet has four possible options to reach the Sink. The farthest sensor node away from the Sink was four hops away. The sensor activity used for training the c-SVM model were taken from experimental results from the Sink-in-the-Middle topology.

TABLE II: SVM Training and Evaluation Data Set when Sink is in the middle

Attack	Training Set - Support Vectors			Evaluation Set		
	Benign	Viral	Total	Benign	Viral	Total
Selective Forward - Block Node	80	80	160	20	20	40
Selective Forward - Forwarding Ratio	80	80	160	20	20	40
Selective Forward & Blackhole - Block Node	80	80	160	20	20	40
Selective Forward & Blackhole - Forwarding Ratio	80	80	160	20	20	40
Sinkhole	80	80	160	20	20	40
Sink in the middle (All attacks)	80	400	800	20	100	200
Sink on top (All attacks)				200	800	1000

To retrieve local sensor activity, two sets of experiments were conducted. The benign scenarios in which all nodes within the network were executing the benign application and the malicious scenarios in which at least one node within the network was malicious.

For the current work we used the attacks from the work of [6] and the RMT tool to collect the data [9]. The data was retrieved from each sensor node besides the Sink node at predefined time intervals called epochs and averaged per the distance of the sensor nodes from the Sink node. Each epoch was set to one (1) second and the total number of epochs used per distance was 25. The benign application was a periodic one and each sensor generated 60 packets per second.

The data taken from the experimental results when the Sink was placed in the middle, was split to 80% and 20% for the training and evaluation stage respectively. The number of epochs used for the training set and the evaluation set are shown in Table II.

A. Training

At the training stage we used routing data taken for the Selective Forward, Blackhole, and Sinkhole attacks in the best case scenario in which the Sink is placed in the middle of the sensor network. In our case we have two classes, the benign and the malicious class, denoted -1 and 1 respectively. The l is the number of epochs of averaged routing layer data of sensors based on the distance from the Sink.

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 our training set
3) Apply the RBF kernel to our training set
4) Apply cross validation and grid search to our training set
5) Input our training set and (C, γ) values to create the SVM model (to derive the optimal hyperplane (see Figure 1))

Based on the attack we use for our training model the significant parameters varied. For Selective Forward attacks, the scaling step returns data for only four parameters: *data packets received*, *data packets forwarded*, *packets dropped* and *announcements received*. For the Sinkhole attack, the scaling step returns the training set for all five parameters shown in Table I. The results of the free parameters (C, γ) are shown in column titled Significant routing parameters in Table III. We also created SVM models for each attack including the non-significant parameters to evaluate the impact of the parameters (shown in column All routing parameters in Table III).

B. Evaluation of SVM training model

From the training stage we created an SVM model with the adjacent (C, γ) that we are going to use for the evaluation stage. The data sets that were used for the evaluation are shown in Table II.

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

1) Label data into benign and malicious in the form of vectors (x_i, y_i)
2) Apply scaling to our training set
3) Input our evaluation sets and (C, γ) values that we retrieve from our training stage to the SVM model

To present our results, we use the Accuracy parameter (ACC) as an indicator for our model. The equation for ACC is shown in (4). We also use the Confusion Matrix that classifies the types of the alarms as shown in Table IV.

$$ACC = \frac{\sum TruePositives + \sum TrueNegatives}{\sum TotalPopulation} \quad (4)$$

The True Positive/Negative values are the correct predictions of the model. True Positive means that the model was given an activity that was benign and was correctly identified. The same applies to True Negative values when given a viral activity and the model correctly identifies it.

The results of our SVM model are shown in Figures 3 and 4. The SF notation stands for Selective Forward, FR is Forwarding Ratio, BN is Block Node, SFBH is Selective Forward and Blackhole attack. Sink-middle scenario, is the scenario in which all data from all malicious and benign scenarios were used in the training stage to derive a general SVM model that can capture routing layer attacks (see in Table II). The general SVM model was evaluated using data taken from the network topology Sink in the middle and data from a different topology, when the Sink is found on top of the network. In the Sink on top scenario, all data from malicious and benign scenarios in the network topology in which the Sink is found on top of the network, were used to evaluate the general SVM model trained with Sink in the middle scenario (see in Table II).

For each attack we created an SVM model and we later evaluated it with its corresponding evaluation set. Figure 3

TABLE III: SVM (C, γ) values from our training results

	All Routing Parameters		Significant Routing Parameters	
Training Set	C	γ	C	γ
Selective Forward - Forwarding Ratio	32.0	0.0078	128.0	0.5
Selective Forward - Block Node	2048.0	0.0049	0.03125	0.5
Selective Forward + Blackhole - Forwarding Ratio	8.0	0.0078	0.03125	0.0078
Selective Forward + Blackhole - Block Node	8.0	0.5	2.0	0.0078
Sinkhole	32.0	0.0078	8.0	2.0
Sink in the middle (All attacks)	0.125	0.0078	0.125	0.0078125

TABLE IV: Confusion Matrix

		True diagnosis		Percentage
		Benign	Viral	Correct
True Condition	Benign	True Positive	False Negative	%
	Viral	False Positive	True Negative	%

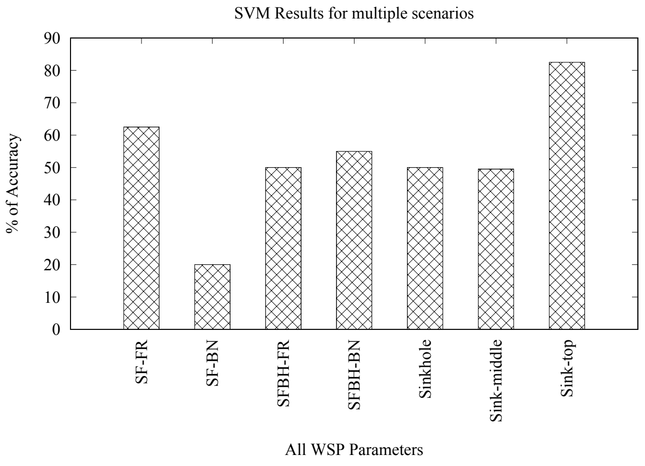


Fig. 3: Percentage of accuracy of SVM model per malicious scenario with all routing parameters

shows the percentage of accuracy of our SVM training model when we used all routing parameters. At the scaling step we included the 0 value parameters to examine the importance of using significant parameters. Figure 4, shows the evaluation results of our training SVM model in which we only used the significant routing parameters. Comparing the two cases the results show that we get higher accuracy levels when only the significant routing layer parameters are used. The highest percentage of attack classification accuracy was 82.5% when all routing layer parameters were used in training stage, whereas by only using the significant routing layer parameters we achieved 100% accuracy.

As shown in Figure 4, SVM managed to get 100% classification for the attacks of Selective Forward along with Blackhole attack and the Sinkhole attack. When Selective Forward is evaluated with either Forwarding Ratio or Block Node, the best accuracy level the SVM model achieves is 87.6%. We also trained our SVM model with the data from all attacks when the Sink is in the middle and evaluated it with the corresponding evaluation set. The SVM model was able to classify the attacks with an accuracy level of 85%. We also use the SVM model that was trained when the Sink is in the middle topology with data from the topology where the Sink is found on top of the grid. The SVM model was able to classify the attacks with 81% accuracy.

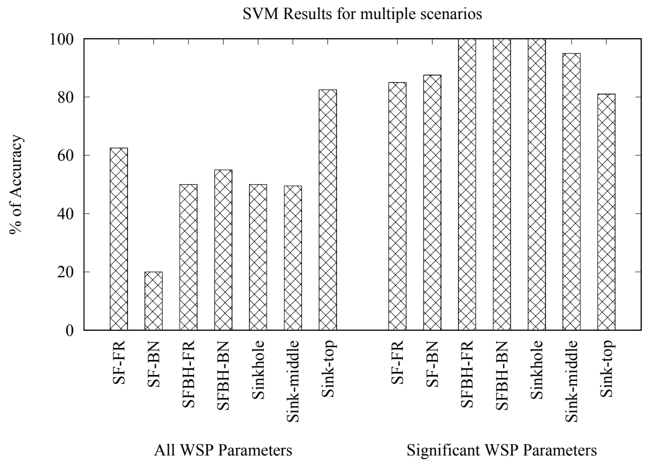


Fig. 4: Percentage of accuracy of SVM model per malicious scenario when using only significant parameters

TABLE V: SVM: Selective Forward - Forwarding Ratio

Evaluation Set		True diagnosis		Percentage
		Benign	Viral	Correct
SF-FR	Benign	15	5	75%
	Viral	1	19	95%

Accuracy ratio(ACC)= 0.85

TABLE VI: SVM: Selective Forward - Block Node

Evaluation Set		True diagnosis		Percentage
		Benign	Viral	Correct
SF-BN	Benign	15	5	75%
	Viral	0	20	100%

Accuracy ratio(ACC)= 0.875

Tables V, VI, VII, VIII, IX, X and XI show the Confusion Matrix for each attack scenario and the accuracy levels that we used in Figure 4. In the Selective Forward - Forwarding Ratio, the accuracy level is 0.85% while the SVM model was able to detect 95% of the malicious behavior but only 25% of the benign behavior (see Confusion Matrix V). In the Selective Forward - Block Node, the SVM model was able

TABLE VII: SVM: Selective Forward and Blackhole - Forwarding Ratio

Evaluation Set		True diagnosis Benign	Viral	Percentage Correct
SFBH-FN	Benign	20	0	100%
	Viral	0	20	100%

Accuracy ratio(ACC)= 1

TABLE VIII: SVM: Selective Forward and Blackhole - Block Node

Evaluation Set		True diagnosis Benign	Viral	Percentage Correct
SFBH-BN	Benign	20	0	100%
	Viral	0	20	100%

Accuracy ratio(ACC)= 1

TABLE IX: SVM: Sinkhole

Evaluation Set		True diagnosis Benign	Viral	Percentage Correct
Sinkhole	Benign	20	0	100%
	Viral	0	20	100%

Accuracy ratio(ACC)= 1

TABLE X: SVM: Evaluation of all routing layer attack when Sink is in the middle of the grid

Evaluation Set		True diagnosis Benign	Viral	Percentage Correct
Sink - middle	Benign	100	0	100%
	Viral	5	15	75%

Accuracy ratio(ACC)= 0.958

TABLE XI: Evaluation of the SVM training model when Sink is in the middle of the grid with node behavior in a topology where the Sink is found on top of the grid

Evaluation Set		True diagnosis Benign	Viral	Percentage Correct
Sink - on Top	Benign	175	25	88%
	Viral	168	635	79%

Accuracy ratio(ACC)= 0.81

to identify correctly all malicious behavior and achieved a 75% percentage correctness to classify benign node behavior. The accuracy level compared to the SF-FR was increased by 0.025% (see Confusion Matrix VI).

C. SVM complexity

SVM models have proven to be good indicators of whether sensor activity is malicious or benign but they come at the cost of high complexity. According to [7], 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 a number of operations proportional to R^3 is required. The cost of complexity increases if the support vectors need to be discovered. The authors in [7] also state that computing the kernel function has a high computation cost as well.

The online analysis computational overhead is also a prohibiting factor to implement SVM on constrained nodes [3], [4]. The work in [3], measures computation overhead based on running time for the training and evaluation procedures. The experiments were conducted in desktop computers with Intel Core I7 3.40 GHz processor. The shortest running time for the evaluation phase was 2.6 seconds and the longest 25.73 seconds. The running time within a constrained node will be in the order of minutes when a sensor node may be equipped with a 8MHz microcontroller [10].

SVM computational complexity overhead makes it difficult, if not impossible, to be implemented in a constrained node. The authors [1] and [11] apply SVM machines in non-constrained nodes to detect possible intervention to the network or data outliers to avoid imposing computational overhead to the constrained nodes.

V. Conclusions

We proposed a c-SVM machine learning model as an anomaly IDS that is trained and evaluated using both benign and malicious local sensor activity. The derived c-SVM model was also evaluated with activity from a different topology than the one used for the training.

The detection accuracy levels reach up to 100% when the Blackhole and Sinkhole attacks were present. The lowest accuracy level was 81% when the model was evaluated in a different network topology for all routing attacks.

The complexity of the c-SVM determines the location of the IDS in the IoT. Based on the computational cost of the c-SVM the IDS should be placed in high energy nodes, such as in the gateway node that connects the IoT to the Internet.

References

[1] S. Kaplantzis, A. Shilton, N. Mani, and Y. Sekercioglu, "Detecting Selective Forwarding Attacks in Wireless Sensor Networks using Support Vector Machines," in *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*, Dec 2007, pp. 335–340.

[2] Enamul Kabir and Jiankun Hu and Hua Wang and Guangping Zhuo, "A Novel Statistical Technique for Intrusion Detection Ssystems," *Future Generation Computer Systems*, 2018.

[3] Abdulla Amin Aburomman and Mamun Bin Ibne Reaz, "A novel weighted support vector machines multiclass classifier based on differential evolution for intrusion detection systems," *Information Sciences*, vol. 414, pp. 225 – 246, 2017.

[4] Sedjelmaci, Hichem and Feham, Mohamed, "Novel Hybrid Intrusion Detection System for Clustered Wireless Sensor Network," *arXiv preprint arXiv:1108.2656*, 2011.

[5] Ioannou, Christiana and Vassiliou, Vasos, "An Intrusion Detection System for Constrained WSN and IoT Nodes Based on Binary Logistic Regression," in *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWIM '18. ACM, 2018.

[6] C. Ioannou and V. Vassiliou, "The Impact of Network Layer Attacks in Wireless Sensor Networks," in *International Workshop on Secure Internet of Things (SIoT 2016)*, Crete, Greece, Sep. 2016.

[7] L. Bottou and C.-J. Lin, "Support Vector Machine Solvers," *Large scale kernel machines*, pp. 301–320, 2007.

[8] C.-C. Chang and C.-J. Lin, "LIBSVM: a Library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[9] C. Ioannou, V. Vassiliou, and C. Sergiou, “RMT: A Wireless Sensor Network Monitoring Tool,” in *Proceedings of the 13th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, ser. PE-WASUN ’16, 2016, pp. 45–49.

[10] *Tmote Sky Ultra Low Power IEEE 802.15.4 compliant wireless sensor module*, Moteiv Corporation, 6 2006.

[11] Y. Zhang, N. Meratnia, and P. Havinga, “Adaptive and Online One-Class Support Vector Machine-Based Outlier Detection Techniques for Wireless Sensor Networks,” in *Proceedings of the 2009 International Conference on Advanced Information Networking and Applications Workshops*, ser. WAINA ’09. Washington, DC, USA: IEEE Computer Society, 2009.