Data-Driven Intrusion Detection for Ambient Intelligence

Ioannis Chatzigiannakis[1(✉)], Luca Maiano[1], Panagiotis Trakadas[2], Aris Anagnostopoulos[1], Federico Bacci[1], Panagiotis Karkazis[3], Paul G. Spirakis[4,5], and Theodore Zahariadis[2]

[1] Sapienza University of Rome, Rome, Italy
ichatz@diag.uniroma1.it, lucamaiano@gmail.com, aris@dis.uniroma1.it, fedeb703@gmail.com
[2] National and Kapodistrian University of Athens, Athens, Greece
{ptrakadas,zahariad}@uoa.gr
[3] University of West Attica, Aigaleo, Greece
p.karkazis@uniwa.gr
[4] Computer Science Department, University of Liverpool, Liverpool, UK
p.spirakis@liverpool.ac.uk
[5] Computer Engineering and Informatics Department, Patras University, Patras, Greece

Abstract. Billions of embedded processors are being attached to everyday objects and houseware equipment to enhance daily activities and enable smart living. These embedded processors have enough processing capabilities to process sensor data to produce smart insights, and are designed to operate for months without the need of physical interventions. Despite the compelling features of Internet of Things (IoT), applied at several home-oriented use cases (e.g., lighting, security, heating, comfort), due to the lack of a physical flow of information (e.g., absence of switches and cable-based gateways), the security of such networks is impeding their rapid deployment. In this work we look into IPv6 based IoT deployments, since it is the leading standard for interconnecting the wireless devices with the Internet and we propose a data-driven anomaly detection system that operates at the transport-layer of 6LoWPAN deployments. We present a comprehensive experimental evaluation carried out using both simulated and real-world experimentation facilities that demonstrates the accuracy of our system against well-known network attacks against 6LoWPAN networks.

1 Introduction to Security Issues and Vulnerabilities in Ambient Intelligence

As IoT devices are becoming integral parts of our homes, a number of issues related to privacy and trust need to be addressed. Data confidentiality, authentication, access control within the IoT network, privacy and trust among users and devices, and the enforcement of security and privacy policies are among these

I. Chatzigiannakis et al. (Eds.): AmI 2019, LNCS 11912, pp. 235–251, 2019.
https://doi.org/10.1007/978-3-030-34255-5_16

issues. However, the different standards and communication stacks involved in combination with the wide variety of embedded hardware components make traditional security countermeasures difficult to be directly applied in the IoT domain. Moreover, the high number of interconnected devices arises scalability issues [28].

The vision of the IoT has led to a competitive market for stakeholders that, in the absence of common standards, marketed proprietary and application-specific solutions, including a variety of hardware platforms, operating systems, communications protocols and data management schemes. Substantial standardisation progress has been made by different bodies (e.g., IETF CoAP, RPL, 6LowPAN, 6TSCH, IEEE 802.15.4e, ETSI M2M, 3GPP MTC, oneM2M), providing technical solutions tailored to the resource-constrained embedded nodes, ranging from the lower to the upper OSI layers. Yet, until now, no standard has managed to attract the vast majority of the stakeholders and dominate the domain. This comes as no surprise as in most cases embedded systems are indeed application-specific and no single protocol stack can address all possible functional and non-functional requirements and specifications [13].

Clearly satisfying the security and privacy requirements of each different connectivity architecture and protocol stack is a complicated task. Moreover, the special operating conditions of IoT networks makes it even more challenging [11]. The wireless devices are more vulnerable to various attacks as their location is not known at design time and protection against tampering is very difficult due to their low cost. Therefore, it is easy to assume that the adversary can easily capture the devices, and easily read the content of their memory; thus learning the cryptographic secrets and possibly modifying their behavior. In addition, the high node-to-human ratio makes it infeasible to even consider the presence of an online trusted server that monitors and maintains individual nodes constantly. Thus techniques of pre-distribution of keys are much less effective than in traditional networks [9]. Furthermore, as sensor nodes are battery operated, security systems must reduce the energy consumption. Also, since sensor nodes have limited computing and storage capability, cryptographic algorithms and protocols that require intensive computation, communication, or storage are simply not applicable in sensor networks [3,8]. It is too costly (in terms of computation) to authenticate using a public key and too costly (in terms of memory and computation) to store one-way chains of keys.

These constraints greatly increase the difficulty of securing IoT networks and make them more vulnerable to security threats. Still, since building secure IIoT networks is of paramount importance, the only viable solution is to combine different techniques for securing the system, i.e., implement secure routing schemes, secure aggregation, provide group key establishment methods, cryptographically encrypt messages etc.; although each single level defense mechanism is highly vulnerable, the combination of multiple attacking angles increases the overall achieved security. Towards this direction, in this paper we focus on the use of intrusion detection systems as an additional mechanism to further improve the security of IIoT networks. In contrast to the other approaches, intrusion

detection can protect from both inside and outside adversaries. Compared to existing systems for IIoT networks, our approach is more complete in the sense that it addresses all the levels of the node stack and also is more energy efficient.

1.1 Security Threats

Unlike in typical stand-alone wireless network deployments, IoT have a different purpose, in the sense that communications will not involve human interaction. This differentiates the challenges for securing the system and therefore the approaches for offering protection need to be reconsidered. There are many publications [5,20,24,31] that consider some of the most significant security problems. In this work we try to summarize all the existing threats and point out the major attacks against IIoT. We call an IIoT node a *normal node* if it operates based on the system specifications. Otherwise, it is a *malicious node* or an *adversary*.

Wireless Threats. The most basic threats are due to the nature of communication that takes place over a wireless channel. Wireless communication suffer from a number of vulnerabilities:

1. **Eavesdropping.** The most easy way is to overhear the information that a node transmits or receives and then analyze the captured data and extract sensitive information without interacting with the network. These attacks are also known as *passive attacks*.
2. **Data alteration.** An adversary can cause collisions of wireless transmissions and then try to modify the message exchanged between wireless parties. These attacks are also known as *active attacks*.
3. **Identity theft.** Untethered in nature, the adversary can impersonate a legitimate user and when the original user is inactive, transmit messages without being noticed.

Routing Threats. The simplicity of many routing protocols for IIoT networks makes them an easy target for attacks. Karlof and Wagner in [21] classify the routing attacks into the following categories:

1. **Spoofed, altered, or replayed routing information.** While sending the data, the information in transit may be altered, spoofed, replayed, or destroyed. Since sensor nodes usually have only short range transmission, an attacker with high processing power and larger communication range could attack several sensors simultaneously and modify the transmitted information.
2. **Selective forwarding.** In this kind of attack a malicious node may refuse to forward every messages it gets, acting as black hole or it can forward some messages to the wrong receiver and simply drop others.
3. **Sinkhole attacks.** In the Sinkhole attack, the goal of the attacker is to attract all the traffic. Especially, in the case of a flooding based protocol

the malicious node may listen to requests for routes, and then reply to the requesting node with messages containing a bogus route with the shortest path to the requested destination.

4. **Sybil attacks.** In Sybil attack the compromised node presents itself with multiple nodes identity. This type of attack tries to degrade the usage and the efficiency of the distributed algorithms that are used. Sybil attack can be performed against distributed storage, routing, data aggregation, voting, fair resource allocation, and misbehavior detection [25].
5. **Wormholes.** Wormhole attack [12] is an attack in which the malicious node tunnels messages from one part of the network over a link, that doesn't exist normally, to another part of the network. The simplest form of the wormhole attack is to convince two nodes that they are neighbors. This attack would likely be used in combination with selective forwarding or eavesdropping.
6. ***HELLO* flood attacks.** This attack is based on the use by many protocols of broadcast *Hello* messages to announce themselves in the network. So, an attacker with greater range of transmission may send many *Hello* messages to a large number of nodes in a wide area of the network. These nodes are then convinced that the attacker is their neighbor. Consequently the network is left in a state of confusion.
7. **Acknowledgment.** Some IIoT network routing algorithms require link layer acknowledgments. A compromised node may exploit this by spoofing these acknowledgments, thus convincing the sender that a weak link is strong or a dead sensor is alive.

Denial of Service (DoS). This class of attacks is not concerned with the information that is transmitted. Rather, the goal of the attacker is to exhaust the resources of the network and cause it not to function properly. Wood and Stankovic [32,33] classify several forms of DoS attacks based on the layer that the attack uses. Some of these were already mentioned so we will not repeat them. At the physical layer the attacks take the form of jamming and tampering [24]. Jamming is done by interfering with the radio frequencies nodes are using. Tampering refers to the physical altering or even damaging of the nodes. An attacker can damage and replace a node, for example, by stealing or replacing information or cryptographic keys. At the link layer the attacker can generate collisions and exhaustion may be caused by protocols that attempt retransmission repeatedly, even when triggered by an unusual and suspicious collision.

1.2 Security Solutions

Several attempts have been made in the past years towards preventing intrusions over IoT networks based on alternative encryption and authentication techniques. Some solutions focus on the validation of the integrity of message exchanges in order to securely route information across the wireless medium [30,36]. Others propose lightweight, distributed secure group communication primitives that operate on-top of the networking layer to protect data and

to cope with potential compromises [10,14], while others work at the application layer and propose energy-efficient encryption methods to guarantee the confidentiality of the information exchanged [6]. However, these approaches cannot completely prevent malicious users from intruding the network and tamper with the operation of the network [7]. For example, the above techniques may not be fully protected against compromised nodes which participate in the network and already have the shared cryptographic keys [35].

In the context of the IP-based IoT solutions, consideration of TCP/IP security protocols is important as these protocols are designed to fit the IP network concept and technology. While a wide range of specialized as well as general-purpose key exchange and security solutions exist for the Internet domain, the 6LoWPAN and CoRE IETF working groups look into IKEv2/IPsec [22], TLS/SSL [27], DTLS [26], HIP [23], PANA [16], and EAP [1] as candidate solutions of IIoT, we focus on the discussion of in this paper. Application layer solutions such as SSH [34] also exist, however, these are currently not considered.

1.3 Anomaly Detection for Internet of Things

In an attempt to improve the overall security of networks, a number of Intrusion detection systems (IDS) have been proposed [6]. Such IDS architectures are classified into two basic categories depending on the *data collection mechanism*:

- **Host-based.** The IDS consults several type of log files (kernel, system, application, etc.) and compares the logs against an internal database of common signatures for known attacks.
- **Network-based.** The IDS is scanning network packets, auditing packet information, and logging any suspicious packets.

IDS architectures can be further classified based on the *detection technique*:

- **Signature-based.** IDS centers on finding an occurrence of predefined signatures or behavior that matches a previously known malicious action or indicates an intrusion.
- **Anomaly-based.** IDS checks for any behaviors that fall outside the predefined or accepted model of behavior.
- **Specification-based.** IDS defines a set of constrains that are indicative of a program's or protocol's correct operation.

Furthermore, *IDS architectures specific to wireless ad-hoc networks* are further divided into three categories:

- **Stand-alone.** Each node operates as a independent IDS and is responsible for detecting attacks only for itself. Such an IDS does not share any information or cooperate with other systems. This architecture implies that all the nodes of the network are capable of running an IDS.
- **Distributed and Cooperative.** All nodes are running their own IDS, but they also cooperate in order to create a global intrusion detection mechanism.

– **Hierarchical.** The network is divided into clusters with cluster-head nodes. These nodes are responsible for routing within the cluster and accept all the accusation messages from the other cluster-members indicating something malicious. Additionally, the cluster-head nodes may also detect attacks against the other cluster-head nodes of the network, as they constitute the backbone of the routing infrastructure.

2 Materials and Method

Existing security solutions cannot completely prevent malicious users from intruding the network and tamper with the operation of the network. In this paper, we look into intrusion detection as a second line of defence to protect IoT once an intrusion is detected by activating certain actions to minimize damages, gather evidence for the prosecution, and even launch counter-attacks. We propose an *Anomaly-based* intrusion detection system (IDS) that operates at the transport-layer level on top of 6LoWPAN networks. Assuming an IPv6 enabled IoT, a solution that operates at transport-layer is generic enough to be applied also in combination with closed systems that implement proprietary hardware and custom network protocols. Our solution does not require any particular implementation of 6LoWPAN or RPL, or any additional encryption primitive.

Our IDS uses the ICMPv6 control messages defined within 6LoWPAN to *passively* collect high-level information for the state of the network, such as the round-trip-time, hop distance and packet loss. The IDS periodically transmits a sequence of ICMPv6 control messages to each node of the network based on the public IP addresses. Then it checks for any behaviours that fall outside the predefined, accepted model of behaviour.

In contrast to existing general purpose security systems, in IoT, it is simply infeasible to expect a network administrator to be aware of the full range of potentially relevant possibilities and be able to pull them together manually. For this reason, we completely avoid defining a series of hard-coded alarm limits associated with assumed "steady states" for the network, known to send a large number of false alarms. Instead, we use a combination of machine-learning and statistical analysis to address the specific problem of anomaly detection.

We assume that there is an initial period of time where the network administrator can collect sufficient data regarding the operation of the network. Then periodically we examine the operation of the network by collecting additional statistics and try to identify observations which differ significantly from the majority of the initial data.

Given a set of traces collected: a *network-level anomaly detector* characterizes if the network is affected by at least 1 malicious node; a *class-sensitive network-level anomaly detector* characterizes if the network is affected by at least 1 malicious node of a specific attack class; a *node-level anomaly detector* characterizes each node of the network if it is a malicious node or not; and a *class-sensitive node-level anomaly detector* characterizes each node of the network if it is a malicious node of a specific attack class. The performance of an anomaly detector

is characterized by the *precision*, that is the ability of the detector not to label as positive a sample that is negative and is measured by the ratio of correctly labelled as L to all items actually labelled as L; the *accuracy*, representing the set of labels detected that exactly match the corresponding set of actual labels; the *sensitivity*, the ability of the detector to find all the positive samples and is measured by the ratio of items correctly labelled as L to all items that belong to label L; and the *overhead*, the number of packets that constitute the trace.

2.1 Experimental Scenario

The target of our experiments aim at providing an automated way of recognizing intrusions in a network. Given an RPL DODAG, we generate ICMP packets for each node of the network. Figure 1 depicts our proposed methodology.

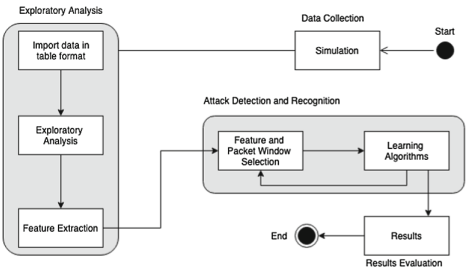


Fig. 1. Methodology flow diagram.

The experiments are based on simulated networks generated by the Cooja tool, the standard Contiki network simulator. We conducted several experiments on different grid schemes (i.e., 3×3, 4×4 and 5×5 grids) as well as random topologies (of 9, 16, 25 nodes). For each node, we collected 200 ICMP packets (sending ping messages from the root node to each other node of the network in every 5 s), the round trip time (RTT) of each of these messages and the distance of the node from the RPL DODAG root. We parse raw data collected from each experiment to extract relevant features in a table structure. This is helpful to gather statistical information from the dataset. From the analysis of this set of data, we expected to retrieve enough information to learn the usual behaviour of the network.

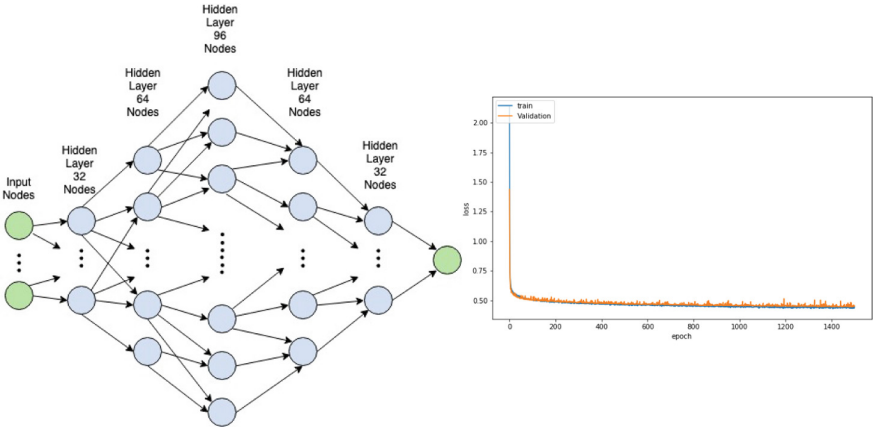
Given the generated network topologies, we substitute some of the nodes of the network with malicious nodes realizing either Black Hole or Grey Hole attacks. The number of malicious nodes is significantly fewer than the number of benign nodes. This is typical in anomaly-detection problems. Since all nodes in the experiment run RPL protocol, we expect that an attacked node will affect the transmissions of its neighbours. Even more, we realized that the malicious nodes usually have more or less important effects on the entire network. Thus all nodes belonging to an attacked network have been labelled as attacked.

After this first exploratory analysis of the data, we select a set of features based on the statistics of the ICMP packets collected over a non-overlapping window of ICMP packets, to train and evaluate the learning algorithms. IMCP packets window is a subset obtained dividing the 200 ICMP messages received by a node by a fixed number in the interval $[12, 24, 48, 100, 200]$ and extract features on that number of ICMP packets. Finally, we run the learning algorithms on the selected dataset and collect results.

2.2 Methods for Anomally Detection

K-Nearest Neighbors (KNN). KNN algorithm is a non-parametric method used for classification and regression [2]. The input consists of the k closest training examples in the feature space. An object is classified by a plurality vote of its neighbours, with the object being assigned to the class that is most common among its k nearest neighbours (k is a positive integer, typically small).

Random Forest (RF). Random Forest (RF) algorithms employ a technique known as bagging, whereby data instances are resampled multiple times to produce multiple training subsets from the training data [4]. Decision trees are then created from each training subset until ensembles of trees have been created. Each tree then casts a unit vote for the outcome of an incoming data instance class label. RF is flexible, with constrained computational resources required.



(a) Neural Network Architecture. (b) Traning and Validation Loss.

Fig. 2. Neural Network model for attack detection.

Deep Neural Network (DNN). Artificial neural networks (ANN) are computing systems that are inspired by the biological neural networks that constitute human brains (Fig. 2). Each neuron performs some calculation and outputs a value that is then spread through all its outgoing connections as input into other units [19]. Connections are characterized by weights that correspond to the importance of the link between two neurons. The computation performed by a unit is separated into two stages: the aggregation and the activation functions. The aggregation function calculates the sum of the inputs received by the unit through all its incoming connections. The resulting value is then fed into the activation function. Neurons are organized in levels. Layers between input and output layers are called the hidden layers. When a new input is given, information passes across layers until the output, where classification happens.

Support Vector Machines (SVM). The basic idea of SVM for time-series approximation is mapping the data into a high-dimensional feature space by a nonlinear mapping and then performing a linear regression in the feature space [29]. The nonlinear mapping can be efficiently computed through a kernel function, without iterating over all the corresponding data points. Given the kernel function, the SVM learner tries to find a hyperplane that separates positive from negative data points and at the same time maximizes the separation (margin) between them. This method is known to be resilient to overfitting and to have good generalization performance, due to the max-margin criterion used during optimization. Furthermore, the SVM is guaranteed to converge to a global optimum due to the corresponding convex optimization formulation.

2.3 Prediction Performance Measure

The performance of the classifiers is based on calculating the precision, accuracy, recall and F1 Score for each class and then computing their unweighted mean. This approach does not take label imbalance into account, e.g., when 95% of items are labelled Normal and 5% are labelled Under-Attack, if all the items are labelled as Normal, the accuracy would be 95% but all items from label Under-Attack would be misclassified. For this reason we made sure that there is a good balance across the different labels considered during both training and validation. The four metrics used are defined as follows:

Precision (or Positive Predictive Value (PPV)) – represents the ratio of items correctly labelled as L to all items actually labelled as L, that is, the ratio $\frac{TP}{(TP+FP)}$, where TP is the number of true positives and FP is the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

Accuracy (or Proportion Correct) – represents the set of labels predicted that exactly match the corresponding set of actual labels.

Recall (or Sensitivity or True Positive Rate or Probability of Detection (PD) or Detection Rate) – represents the ratio of items correctly labelled as L to

all items that belong to label L, that is, the ratio $\frac{TP}{(TP+FN)}$, where TP is the number of true positives and FN is the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

F1 score, or balanced F-score or F-measure represents a weighted average of the precision and recall, where 1 is the best score and 0 the worst score. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is: $\frac{2\times(precision\times recall)}{(precision+recall)}$.

AUC (or Area Under the ROC curve) – ROC is a probability curve and AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes.

2.4 Implementation Details

The learning algorithms are implemented using Python 3.7.1 in combination with Keras 2.2.4, Pandas 0.24.1, Scikit 0.20.3 and Numpy 1.15.4. All the experiments have been performed on a Macbook Pro 2017 - 2, 3 GHz Intel Core i5, 8 GB 2133 MHz LPDDR3. We release all the dataset and the code used for the experiments on a Github repository (https://github.com/ichatz/iot-netprofiler).

All mentioned algorithms have been extensively used by the state of the art solutions in order to solve anomaly-detection problems [15]. We used the following operating parameters:

1. **K-Nearest Neighbor** (KNN) - using $K = 3$.
2. **Random Forests Classifier** - building 100 estimators.
3. **Support Vector Machines** (SVM) - using a linear kernel.
4. **Deep Neural Network** Classifier - using 5 hidden layers of 32, 64, 96, 64, 32 neurons respectively. Dropout and L2 regularization are applied to reduce overfitting. We use the cross-entropy loss function to evaluate training and validation losses.
5. **K-Means** - with K equal to the number of classes. In this case, data are pre-processed using a PCA.

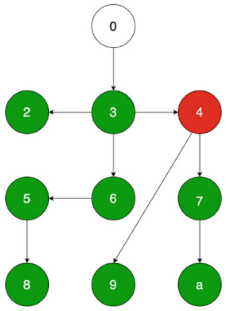
3 Results

3.1 Exploratory Analysis

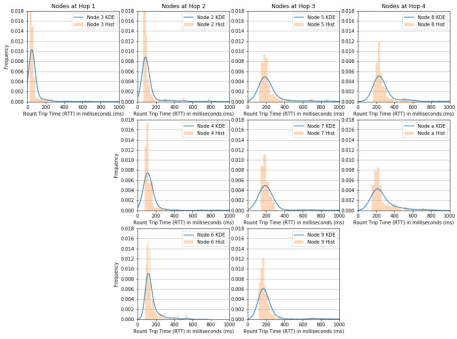
We start with by conducting a first exploratory analysis of the network statistics. Figure 3 depicts the distribution of round-trip-time (RTT) for each node, given the hop-distance from the root, for a topology involving 9 nodes in a 3×3 grid. Similar results are acquired for larger topologies. We keep in mind that the RPL DODAG is designed to have a different configuration each time an RPL instance on the network is running, thus, even with the same topology, the network could have a different structure. However, regardless of the RPL DODAG operating parameters, by simply observing the basic network statistics, it is clear that the distribution of the RTT at each hop-distance is more or less the same. Given the

distribution of the RTT of a node at certain hop-distance from the root, we can usually understand if the node (or one of its neighbours) has been attacked. In fact, depending on which kind of attack has been performed, it is easy to notice that the distribution of a certain number of nodes is affected. In particular, we can distinguish between the following cases:

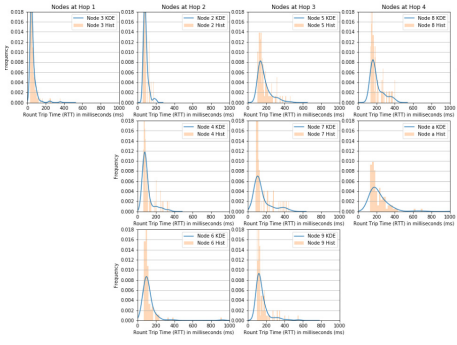
– if the network has been attacked with a Black Hole attack at a non-leaf node N, this attack also affects the children nodes of N. These nodes either remain unavailable for the entire experiment, or after a given period of time they manage to identify an alternative path connecting them with the root node;
– if a Gray Hole attack has been performed, we can still observe an effect on variance and latency values in some nodes.



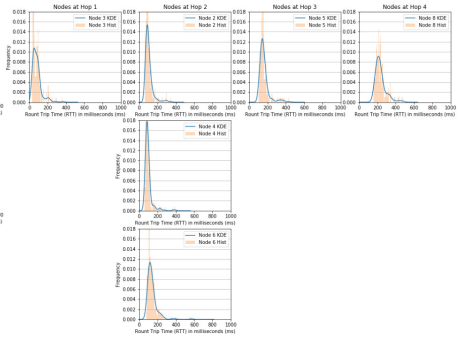
(a) Network Topology.



(b) Normal behaviour.



(c) Grey Hole Attack.



(d) Black Hole Attack.

Fig. 3. RTT distribution of 3×3 topology.

Since the RTT varies over time, it is important to study if there is an *Auto-correlation*, highlighting in this way the similarity between observations as a

function of the time lag between them. Studying *autocorrelation* can also identify possible *seasonability* in the measurements, i.e., periodic fluctuations. Assuming that the distribution of each variable fits a Gaussian distribution, we can use the *Pearson's correlation coefficient* to summarize the correlation between the variables. Pearson's correlation coefficient is a number between −1 and 1 that describes a negative or positive correlation, respectively. A value of zero indicates no correlation. We can calculate the correlation for time-series observations with previous time steps, called lags. Because the correlation of the time series observations is calculated with values of the same series at previous times, this is called a serial correlation, or an autocorrelation. A plot of the autocorrelation of a time series by lag is called the AutoCorrelation Function, or the acronym ACF.

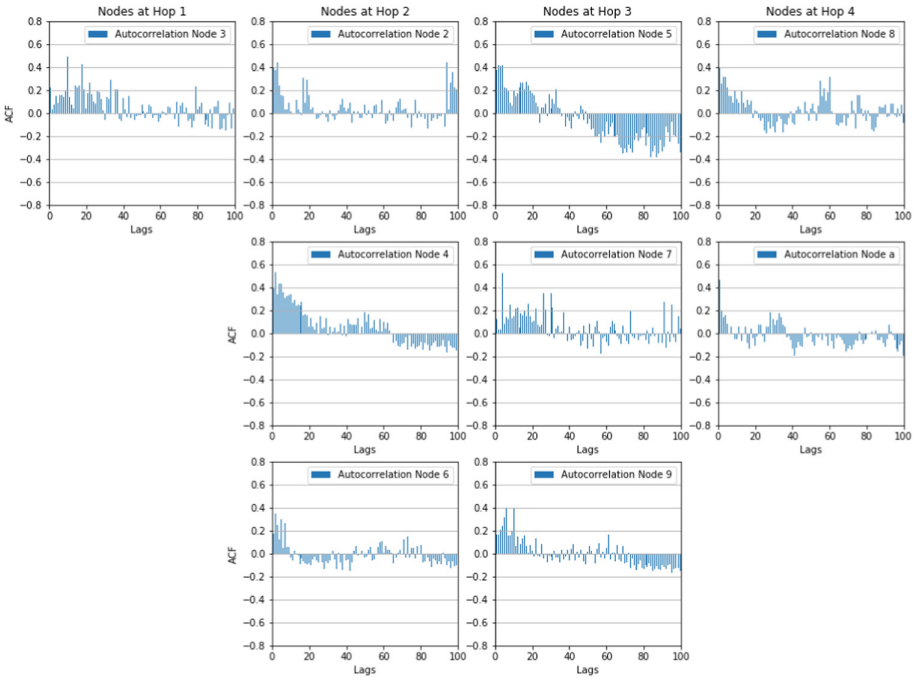


Fig. 4. Autocorrelation of a 3×3 grid experiment.

The results are included in Fig. 4, obtained above do not show particular trends, suggesting that the RTT time series are stationary. In order to make strong assumptions about our data, we computed *Advanced Dickey-Fuller test* [17]. This is a statistical test called a unit root test. The intuition behind a unit root test is that it determines how strongly a time series is defined by a trend. It uses an autoregressive model and optimizes an information criterion across multiple different lag values. The null hypothesis of the test is that the

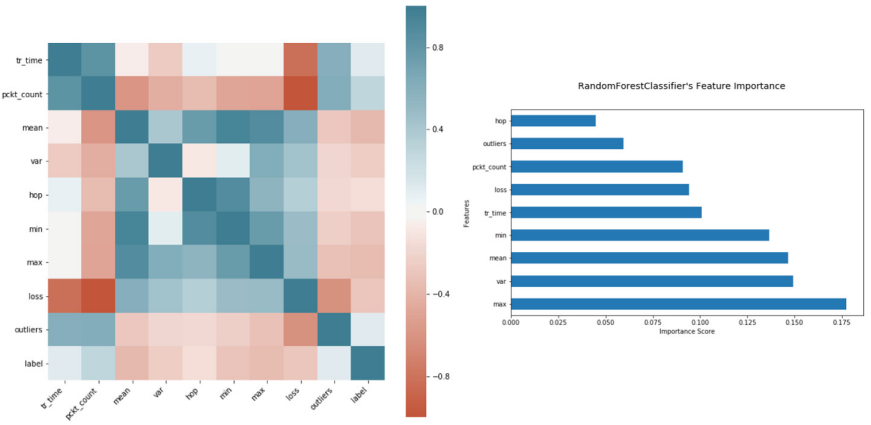
time series can be represented by a unit root, that it is not stationary (has some time-dependent structure). The alternate hypothesis (rejecting the null hypothesis) is that the time series is stationary.

1. Null Hypothesis (H0): If failed to be rejected, it suggests the time series has a unit root, meaning it is non-stationary. It has some time-dependent structure.
2. Alternate Hypothesis (H1): The null hypothesis is rejected; it suggests the time series does not have a unit root, meaning it is stationary. It does not have a time-dependent structure.

Results from Advanced Dickey-Fuller test confirms that the hypothesis that RTTs time series is stationary, at least for more than 79% of nodes. Starting from these results, a *log transform* has been used to flatten out non-stationary nodes back to a linear relationship. This could help learning algorithms; in fact, stationary time series can be easier to model.

3.2 Features Extraction and Selection

We now move one to studying the importance of different features in terms of the learning rate of machine learning and deep learning algorithms. Over a fixed window of packets, we consider the mean, variance, min, and max values of RTT, the number of **outliers** on the RTT value, the hop distance of the node from the root, and the number of lost packets. Remark that since data resulting from different networks could have different mean and variance due to their different network topologies, we apply *feature normalization*.



(a) Correlation Matrix.

(b) Random Forest Classifier Features Importance.

Fig. 5. Features correlation and importance.

Some features could be more important than others, so if we select the right set of features, this could help to improve the results given by a learning classifier. In order to select the best set of features, we calculate the correlation matrix of features. We use the Pearson Coefficient that assigns a value within 1 (i.e. positive correlation) and −1 (negative correlation). Following the approach proposed by Furkan Yusuf Yavuz, Devrim ünal and Ensar Gul in [18], we use *Random Forest Classifier* to iteratively select the most relevant features. In fact, as they suggest, if feature importance is high, it dilutes the effect of the others and may cause overfitting, while less important could slow down (or even deviate) the learning process (Fig. 5).

3.3 Experimental Results

We now examine the performance of different machine learning algorithms to accurately detect attacks in an IoT network. We use two different experimental scenarios. In the first one, we want to detect if a network has been attacked or not. In the second scenario, we try to recognize what kind of attack has been performed if any. In each scenario, we train the learning algorithms with the following approach:

1. Select a window of N ICMP packets for each node ($N \in [12, 24, 48, 100, 200]$).
2. Select a subset of features.
3. Split the dataset using 80% of the data for training and 20% for testing.
4. Train the algorithm using 5-fold cross-validation. Therefore, split the training set to and iteratively evaluate the results on 20% of this set.
5. Finally, we test the algorithm on the test set.

The results suggest that we can accurately detect if an attack has been performed. Attacks are accurately detected by KNN, Random Forests and Deep Neural Network with an accuracy of 80%. Due to the observed stationarity, varying the size of the windows does not affect the results. Even with a window size of 12 packets, which corresponds to 1 min, we can accurately detect an attack in the network. Figure 6 shows the results of this experiment for attack detection.

model	n_classes	accuracy	precision	recall	f1-score	auc roc
knn	2	0.803406	0.797810	0.794549	0.796010	0.794549
random forest	2	0.829721	0.829888	0.815789	0.820818	0.815789
svm	2	0.752322	0.755062	0.726316	0.731972	0.726316
neural network	2	0.823529	0.847324	0.795865	0.806430	0.795865
kmeans	2	0.375891	0.383400	0.382168	0.375668	0.382168

Fig. 6. Attack detection results.

We also examine the classification of specific type of attack. Here KNN and Random Forests achieve the highest accuracy in the range of 71% and 75% accuracy. Figure 7 indicates the different values achieved for each different method considered.

model	n_classes	accuracy	precision	recall	f1-score	auc roc
knn	3	0.712963	0.698080	0.708799	0.701690	0.777039
random forest	3	0.750000	0.759638	0.727007	0.740593	0.791564
svm	3	0.616667	0.686075	0.503812	0.514424	0.628717
neural network	3	0.505556	0.168519	0.333333	0.223862	0.500000
kmeans	3	0.439585	0.433131	0.381860	0.376426	0.538705

Fig. 7. Attack recognition results.

4 Conclusions and Future Work

Our study presents an approach which can detect routing attacks based on a simple analysis of ICMP packets. Packet-drop attacks (black-hole attack and grey-hole attack) are successfully detected by our proposed attack detection models. Our methodology proposes a simple and efficient solution to detect if such attacks have been performed. Experimental results suggest that even kind of attack could be detected with high accuracy if learning models are trained with a sufficiently wide range of data samples. The lack of open-source datasets and the difficulty in collecting this type of data surely makes it difficult to train learning algorithms efficiently. As future work, we intend to extend the dataset with additional attack types and scenarios, introducing a different rate of malicious and normal nodes as well as a larger number of nodes. We also plan to test more sophisticated Neural Network architectures like LSTM, that could help to achieve better results on a larger dataset.

References

1. Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., Levkowetz, H.: Extensible authentication protocol (EAP) (2005)
2. Altman, N.S.: An introduction to kernel and nearest-neighbor nonparametric regression. Am. Stat. **46**(3), 175–185 (1992)
3. Boukerche, A., Chatzigiannakis, I., Nikoletseas, S.: Power-efficient data propagation protocols for wireless sensor networks. Simulation **81**(6), 399–411 (2005)
4. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
5. Butun, I., Morgera, S.D., Sankar, R.: A survey of intrusion detection systems in wireless sensor networks. IEEE Commun. Surv. Tutorials **16**(1), 266–282 (2014)
6. Chatzigiannakis, I., Pyrgelis, A., Spirakis, P.G., Stamatiou, Y.C.: Elliptic curve based zero knowledge proofs and their applicability on resource constrained devices. In: 2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems, pp. 715–720, October 2011

7. Chatzigiannakis, I., Strikos, A.: A decentralized intrusion detection system for increasing security of wireless sensor networks. In: 2007 IEEE Conference on Emerging Technologies and Factory Automation (EFTA 2007), pp. 1408–1411, September 2007
8. Chatzigiannakis, I., Kinalis, A., Nikoletseas, S.: An adaptive power conservation scheme for heterogeneous wireless sensor networks with node redeployment. In: Proceedings of the Seventeenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, pp. 96–105. ACM (2005)
9. Chatzigiannakis, I., Konstantinou, E., Liagkou, V., Spirakis, P.: Design, analysis and performance evaluation of group key establishment in wireless sensor networks. Electron. Notes Theor. Comput. Sci. **171**(1), 17–31 (2007)
10. Chatzigiannakis, I., Konstantinou, E., Liagkou, V., Spirakis, P.: Design, analysis and performance evaluation of group key establishment in wireless sensor networks. Electron. Notes Theor. Comput. Sci. **171**(1), 17–31 (2007). Proceedings of the Second Workshop on Cryptography for Ad-hoc Networks (WCAN 2006)
11. Chatzigiannakis, I., Mylonas, G., Vitaletti, A.: Urban pervasive applications: challenges, scenarios and case studies. Comput. Sci. Rev. **5**(1), 103–118 (2011)
12. Hu, Y., Perrig, A., Johnson, D.B.: Wormhole detection in wireless ad hoc networks. In: Ninth International Conference on Network protocol (ICNP), vol. 1 (2002)
13. Dimitrios, A., Vasileios, G., Dimitrios, G., Ioannis, C.: Employing internet of things technologies for building automation. In: Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012), pp. 1–8. IEEE (2012)
14. Wenliang, D., Deng, J., Han, Y.S., Varshney, P.K., Katz, J., Khalili, A.: A pairwise key predistribution scheme for wireless sensor networks. ACM Trans. Inf. Syst. Secur. (TISSEC) **8**(2), 228–258 (2005)
15. Hassan, S.A., Hussain, F., Hussain, R., Hossain, E.: Machine learning in IoT security: current solutions and future challenges. arXiv:1904.05735v1 (2019)
16. Forsberg, D., Ohba, Y., Patil, B., Tschofenig, H., Yegin, A.: Protocol for carrying authentication for network access (PANA) (2008)
17. Fuller, W.A.: Introduction to Statistical Time Series, 2nd edn. Wiley, Hoboken (1995)
18. ÜNAL, D., GÜL, E., YAVUZ, F.Y.: Deep learning for detection of routing attacks in the internet of things. Int. J. Comput. Intell. Syst. **12**(1), 39–58 (2018)
19. Gamboa, J.C.B.: Deep learning for time-series analysis. arXivpreprint arXiv:1701.01887 (2017)
20. Heer, T., Garcia-Morchon, O., Hummen, R., Keoh, S.L., Kumar, S.S., Wehrle, K.: Security challenges in the IP-based internet of things. Wirel. Pers. Commun. **61**(3), 527–542 (2011)
21. Karlof, C., Wagner, D.: Secure routing in wireless sensor networks: attacks and countermeasures. In: Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, pp. 113–127. IEEE (2003)
22. Kaufman, C.: Internet key exchange (IKEv2) protocol (2005)
23. Moskowitz, R., Nikander, P., Jokela, P., Henderson, T.: Host identity protocol version 2(HIPv2) (2015)
24. Mpitziopoulos, A., Gavalas, D., Konstantopoulos, C., Pantziou, G.: A survey on jamming attacks and countermeasures in wsns. IEEE Commun. Surv. Tutorials **11**(4), 42–56 (2009)
25. Newsome, J., Shi, E., Song, D., Perrig, A.: The sybil attack in sensor networks: analysis & defenses. In: Third International Symposium on Information Processing in Sensor Networks, IPSN 2004, pp. 259–268. IEEE (2004)

26. Phelan, T.: Datagram transport layer security (DTLS) over the datagram congestion control protocol (DCCP) (2008)
27. Rescorla, E.: The transport layer security (TLS) protocol version 1.3 (2018)
28. Sadeghi, A.-R., Wachsmann, C., Waidner, M.: Security and privacy challenges in industrial internet of things. In: 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1–6. IEEE (2015)
29. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, Berlin (2013). https://doi.org/10.1007/978-1-4757-3264-1
30. Velivasaki, T.-H.N., Karkazis, P., Zahariadis, T.V., Trakadas, P.T., Capsalis, C.N.: Trust-aware and link-reliable routing metric composition for wireless sensor networks. Trans. Emerg. Telecommun. Technol. **25**(5), 539–554 (2014)
31. Wallgren, L., Raza, S., Voigt, T.: Routing attacks and countermeasures in the RPL-based internet of things. Int. J. Distrib. Sens. Netw. **9**(8), 794326 (2013)
32. Wood, A.D., Stankovic, J.A.: Denial of service in sensor networks. Computer **35**(10), 54–62 (2002)
33. Wood, A.D., Stankovic, J.A.: A taxonomy for denial-of-service attacks in wireless sensor networks. In: Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems, pp. 739–763 (2004)
34. Ylonen, T., Lonvick, C.: The secure shell (SSH) protocol architecture (2006)
35. Zhang, Y., Lee, W., Huang, Y.-A.: Intrusion detection techniques for mobile wireless networks. Wirel. Netw. **9**(5), 545–556 (2003)
36. Zhou, L., Haas, Z.J.: Securing ad hoc networks. IEEE Netw. **13**(6), 24–30 (1999)