Hindawi
Security and Communication Networks
Volume 2018, Article ID 6063456, 10 pages
https://doi.org/10.1155/2018/6063456

Research Article

A Smart Trust Management Method to Detect On-Off Attacks in the Internet of Things

Jean Caminha,[1,2] **Angelo Perkusich**,[2] **and Mirko Perkusich**[2]

[1]*Computing Institute, Federal University of Mato Grosso, Cuiabá, MT, Brazil*
[2]*Embedded Systems and Pervasive Computing, Federal University of Campina Grande, Campina Grande, PB, Brazil*

Correspondence should be addressed to Jean Caminha; jean@ic.ufmt.br

Received 24 November 2017; Revised 7 February 2018; Accepted 25 February 2018; Published 15 April 2018

Academic Editor: Ilsun You

Internet of Things (IoT) resources cooperate with themselves for requesting and providing services. In heterogeneous and complex environments, those resources must trust each other. On-Off attacks threaten the IoT trust security through nodes performing good and bad behaviors randomly, to avoid being rated as a menace. Some countermeasures demand prior levels of trust knowledge and time to classify a node behavior. In some cases, a malfunctioning node can be mismatched as an attacker. In this paper, we introduce a smart trust management method, based on machine learning and an elastic slide window technique that automatically assesses the IoT resource trust, evaluating service provider attributes. In simulated and real-world data, this method was able to identify On-Off attackers and fault nodes with a precision up to 96% and low time consumption.

1. Introduction

The Internet of Things (IoT) connects people and things to information systems via smart devices based on Internet-oriented devices and knowledge paradigms. In the near future, over 50 billion devices will be connected to the Internet, supporting several applications like smart cities, smart houses, supply chain, and precision agriculture [1]. Given this heterogeneous and complex environment, security is a key concern in the IoT, and devices must be smart to protect systems from threats [2].

The swarm concept applied to the IoT leverages the independent cooperation of devices to execute tasks and demands special infrastructure to support heterogeneous device interactions, such as those in smart cities environments (Figure 1). The components of a swarm system must connect seamlessly and trust each other.

Security and privacy problems challenge the IoT vision. Large amount of data will be produced from billions of interactions between devices and people in new and existing paradigms like cloud computing, machine to machine, Internet of Vehicles, Internet of Energy, and Internet of Sensors [3].

IoT resources interact with themselves by requesting and providing services, sometimes opportunistically as they come into contact with each other. In this context, misbehaving devices may perform discriminatory attacks based on trust abuse. To maximize system security, it is important to evaluate the trustworthiness of service providers in IoT environments [4].

An IoT device can act as a service provider or service requester. A service requester wants to select the best service provider and trust it. A malicious provider resource can offer bad services and information, thereby compromising systems. Trust attacks and their countermeasures are an open issue being addressed by researchers [5].

On-Off attacks are considered a selective attack type. Multiservice IoT architectures may suffer attacks from malicious nodes that perform actions based on type of service they provide to other nodes in the network. A malicious device can provide good and bad services randomly to avoid being rated as a low trust node. An On-Off (OA) attacker can also behave differently with different neighbors to achieve inconsistent trust opinions of the same node. This kind of attack is hard to detect using traditional trust management schemes [6].

Figure 1: The swarm concept applied to IoT enabled smart city.

In addition, not all misbehaving devices are attackers. Some may be devices in malfunction status. Separating attackers nodes from broken nodes is useful for systems administrators to recover the IoT systems.

To mitigate threats like OA, artificial intelligence and machine learning boost the performance of security solutions in IoT architectures. Researches in security using artificial intelligence range from simple modeling attack patterns to sophisticated anomaly detection schemes, which can detect unknown attacks. Machine learning applications are developing security solutions like antivirus, network intrusion systems, and fraud-detection systems [7].

The solution described in this study aims to provide a real-time method to automatically assess OA resources, evaluating service attributes such as data provision or quality-of-service data. For this purpose, we developed an elastic slide window and machine learning based trust management method to aid systems and users to protect themselves against OA.

The main contributions of this study are the following:

(i) Introduction of a method to improve security in IoT by identification of On-Off trust attacks, using less data

(ii) An efficient method to differentiate attackers nodes from broken nodes

(iii) A flexible implementation, which combines machine learning and elastic sliding window to correctly understand variations in trusted devices outputs

(iv) Design and execution of a proof of concept using simulated and real data from a smart city project, demonstrating the efficiency of the method.

This article is structured as follows. Section 2 presents related works on OA. Section 3 presents our smart trust management method and methodologies to validate the solution. Section 4 presents the results achieved in simulated and real-world scenarios. Finally, Section 5 presents our final conclusions and future works.

2. Related Works

Trust evaluation is an essential part of a trust management scheme. There are many different methods to compute the degree of trust in distributed networks. They can be divided into direct and indirect trust. Direct trust refers to methods that infer a trust score owing to direct data observations. Indirect trust uses reputation and recommendations by other peers. Trust scores can persist in a known central node or an authorized third party. In a decentralized model of trust evaluation, a node computes a trust value for every node interaction.

The distributed management approach in [8] computes the trust value locally by nodes. The trust value is based on direct observations, through the service availability of related node. This scheme is time and resource consuming. It required 120 min to fill the local table with suspicious and trusted nodes. Another drawback of this approach is that it did not consider the initial trust level of a peer. A recent study [6] found this introduced reward and punishment scheme as only method to protect against OA menace in IoT environments.

The adaptive security model in [9] is based on a trust evaluation method composed of three complementary components: experiences, observations, and recommendations. It focuses on reducing resource consumption in mobile ad hoc network. The clustering architecture in [10] addresses trust management in the IoT based on the similarity of interest in each cluster. Its prediction mechanism uses the Kalman filter to estimate the trust value in advance.

The trust-based offloading method for mobile M2M communications in [11] uses reinforcement learning and builds a feedback system. The trust level of an initiator node toward other nodes is updated after each communication to enable the node to more precisely evaluate new interactions. It focuses on trust which can improve the energy consumption and computation speed of devices and improve the availability of the system. However, this scheme does not consider the different services provided by a peer, and nor does it consider the trustworthiness of the collected trust data of each node.

RealAlert is a policy-based secure and trustworthy sensing scheme proposed in [12]. In this scheme, the data trustworthiness and IoT node attributes are assessed using anomalous IoT data and contextual information that represents the environment from which anomalous IoT data were obtained. Policy rules are defined to specify how to evaluate the trustworthiness in different situations. New devices or new normal observations may be considered attacker by an outdated policy.

The quantitative model of trust value based on multi-dimensional decision attributes in [13] uses the monitored direct trust value measured from network communication. Packet forwarding capacity, repetition rate, consistency of the packet content, delay, and integrity are evaluated. It adopts the D-S theory to computes the trust. Its drawback is related to a large amount of data collected from various devices in the IoT environment. The amount of data increases exponentially, and continuously streaming data is difficult to manage with traditional network communication data analysis methods.

The study [14] uses a trust relationship scheme based on clustered wireless sensor networks (M2M). A cloud model implements the conversion between qualitative and quantitative data of sensor nodes (trust metrics). To calculate the trustworthiness of sensor nodes, the proposed method considers communication, message, and energy factors, a

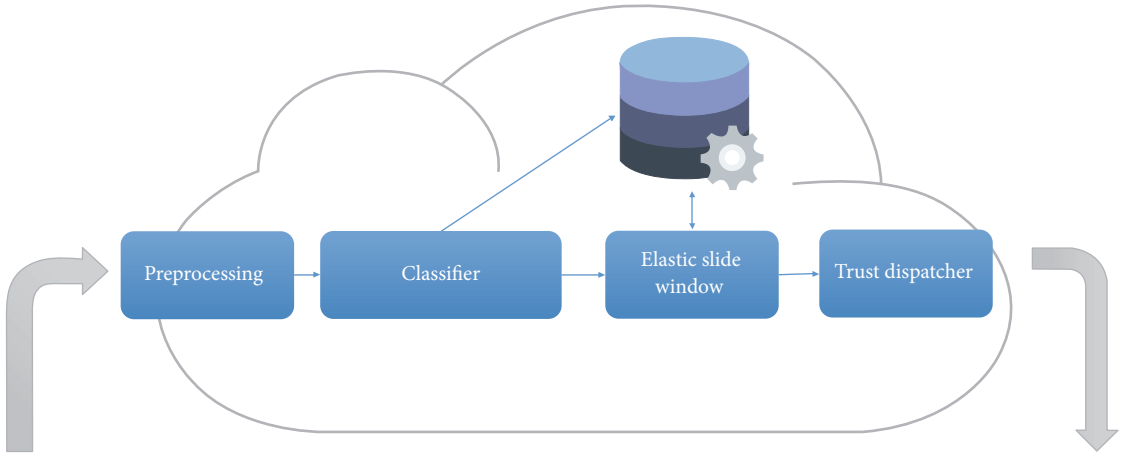

Figure 2: The smart trust management method data flow on cloud.

Table 1: Related works comparison.

Study	FDN	NPTS	DAM
[8]	x		
[9]	x		
[10]	x	x	
[11]	x	x	
[12]			
[13]	x		
[14]			
[15]	x		
Our method	X	X	X

mathematical assignment of a dynamic weight for each trust factor to detect attacks. Cloud solutions are threatened by vendor related problems (proprietary protocols or end-to-end designs), low end devices computer power, and service discovery incompatible methods.

The trust management and redemption scheme in [15] discriminates between temporary errors and malicious behaviors to detect and defend against OA. It uses predictability trust, computed as the ratio of good behavior to the total behavior in the system, and a static sliding window that records previous behavior history. This scheme needs time to compute the entire system behavior. Maintaining a static behavior record, it cannot accommodate new trusted acts.

All studies evaluated were also compared (Table 1) in terms of information (metadata entries) needed to compute the trust score (FDN), no need to know previous trust score from a neighbor node (NPTS), or the differentiation between attacker nodes and malfunctioning (broken) ones (DAM).

The method presented in this work fills the FDN, PTS, and DAM gaps present in other works and aggregates more efficiency in terms of detection precision, recall, and implementation flexibility.

3. Methodology

In this section we demonstrate our proposed method to detect OA in IoT. The two validation scenarios are described. To find the best machine learning classifier, we also conduct a comparison between supervision methods.

3.1. A Smart Trust Management Method to Detect On-Off Attacks in the Internet of Things. The goal of the proposed method is to collaborate with IoT systems to identify OA attacks and broken nodes. Interactions among IoT devices are evaluated using available metadata attributes. This smart trust management method is designed to be accessed through a representational state transfer (REST) application programming interface (API). Figure 2 illustrates the metadata flow of the cloud.

An IoT object metadata can be sent to the method to be evaluated. In the preprocessing phase, data are submitted to related feature type extraction process. Text data are processed by the HashingVectorizer [16]. This process converts text (n-grams only from text inside word boundaries) to a matrix of token occurrences and finds a token string name for the feature integer index mapping. This approach was selected because it does not store a vocabulary dictionary in memory and can also be used in streaming (partial fit).

Each kind of preprocessed data is submitted to a specific machine learning classifier to identify its class. Whenever the data, such as the annual temperature range for a city, is on the range of accepted values it is assumed to be trusted (Figure 3). Otherwise, out-of-range values are assigned as outliers.

The classifier confirms whether it identified a class and returns a decision function value. The decision function is used by our method to determine the elastic slide window size. It is calculated by observing the evaluated data sample distance hyperplane of the model decision function. The degree of separation achieved by the hyperplane has the largest distance to the nearest training data points of any class (the functional margin). A high (or positive) decision function value corresponds to the prediction assurance.

Figure 3: Expected range of trusted values.

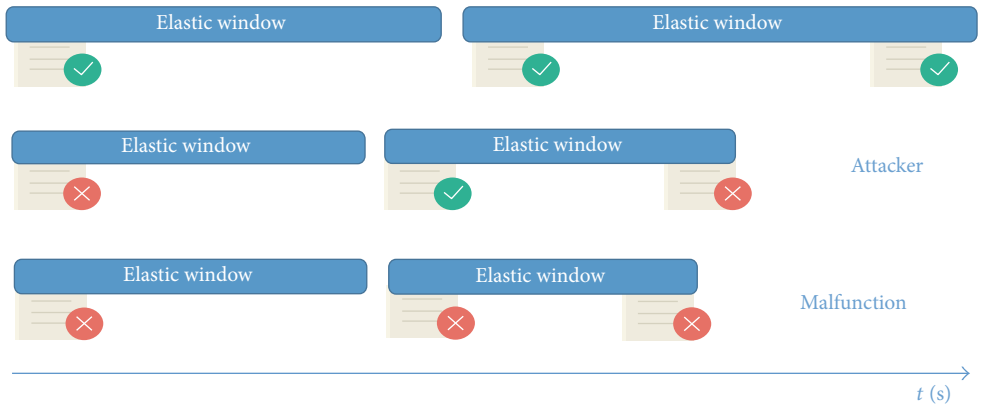

Figure 4: The elastic slide window.

Table 2: Output decisions for two analyses in the same elastic slide window.

Elastic slide window						Output	
	Read 1			Read 2			
Class	High	Low	Class	High	Low	Thing	ESW size
x	x		x	x		Trusted	Decrease
x	x		x		x	Trusted	Increase
x		x	x	x		Trusted	Decrease
x		x	x		x	Trusted	Increase
x	x			x		Attacker	Decrease
x	x				x	Attacker	Increase
x		x		x		Attacker	Decrease
x		x			x	Attacker	Increase
	x		x	x		Attacker	Decrease
	x		x		x	Attacker	Increase
		x	x	x		Attacker	Decrease
		x	x		x	Attacker	Increase
	x			x		Broken	Decrease
	x				x	Broken	Increase
		x		x		Broken	Decrease
		x			x	Broken	Increase

The elastic slide window (ESW) presented in Figure 4 is an important phase of the data flow. It enhances the trust using time frame analysis. An OA sends good and bad read values in a discretionary manner. Healthy systems expect only good values over time. When the classifier sends an identified class and low decision function value, our method assumes there are doubts about trust, and the evaluated resource needs to be tested again in a larger time frame. With each interaction, the elastic slide window is aggregated by decision function values. Low decision function values lead to higher elastic slide windows and permit the method to analyze the variance in significant node behaviors (good or bad).

The trust dispatcher is responsible for saving the elastic slide window size value in a database or memory for future reference. It also determines the trust resource type: Good, On-Off attacker, or Broken. The trust dispatcher implements the decisions mapped in Table 2.

Some implementation details must be considered in our smart trust management method (Algorithm 1). Two variables need to be initialized to the elastic slide window:

```
    input: A metadata m (ID, read)
    output: A predicted type: (Tusted, On-OffAttacker, Broken)
(1) eswAlpha ← alpha;
(2) eswInit ← beta;
(3) NewPrediction ← Classifier.Predict(m);
(4) NewDecisionFunction ← Classifier.DecisionFunction(m);
(5) if m in Database then
(6)     m.SlideWindow ←
          (eswInit + time()) − NewDecisionFunction;
(7)     m.prediction ← NewPrediction;
(8)     if m.SlideWindow ≥ Time() then
(9)         if NewPrediction == −1 and m.prediction == −1 and
              NewDecisionFunction ≤ eswAlpha then
(10)            m.prediction ← 0;
(11)        end
(12)        if NewPrediction != m.prediction and
              NewDecisionFunction ≥ eswAlpha then
(13)            m.prediction ← −1;
(14)        end
(15)        if NewPrediction != m.prediction and
              NewDecisionFunction ≤ eswAlpha then
(16)            m.prediction ← m.prediction;
(17)        end
(18)    end
(19) end
(20) m.SlideWindow ← m.SlideWindow − NewDecisionFunction;
```

Algorithm 1: The smart trust management method algorithm.

eswAlpha(line 1) and *swInit*(line 2). *eswAlpha*(1) allows system administrator to define the value to be considered as trusted in a decision function score (lines 11, 14, and 37). It is also utilized to calculate the growth degree of an ESW (line 30). The variable *swInit* records the initial ESW time (in seconds) for a new resource.

The verification in line 9 checks if a ESW for a resource is greater than the actual computer time and must be considered in this analysis. New ESW sizes are calculated using previous ESW values minus the new decision function value. Negative decision function values aggregate the ESW size (line 30). The output decisions for two analyses in the same elastic slide window are implemented in lines 7, 11, 14, and 17.

The smart trust management method returns results by a REST/API query. Figure 5 shows an example of an output analysis from our method. The smart trust management server is consulted via the Constrained Application Protocol (CoAP) about an object and returns the results in JSON formatted data. For example, the object *ID*: 14 with a metadata payload 46 (degree Celsius) is marked as an *On-Offattacker*. A trust score (the decision function value) is also presented.

Other possible results are *Good* for predictable devices or *Broken* for two or more fault values in the same elastic slide window.

The classifier model used by the smart trust management method was created by the OneClassSVM (RBF kernel, upper and lower bound on fraction of training errors = 0.1, and kernel coefficient = 0.1) method, trained with 2000 samples, implemented with Python 3.6 scikit-learn and LIBSVM

Figure 5: Example of an output result by the smart trust management server.

library [16], version 0.18.0, on a desktop machine with a Core i7 3.60 GHz processor and running Windows 10.

To validate our solution, we prepared two experimental setups: computer simulation and a real-world scenario. The setups have annotated data for OA nodes and broken nodes.

3.2. Simulation Validation. Our simulation setup consists of 51 nodes' set (Figure 6). A node is the destination object representing the data consumer and uses our method to identify trusted and misbehaved nodes. Forty nodes act as

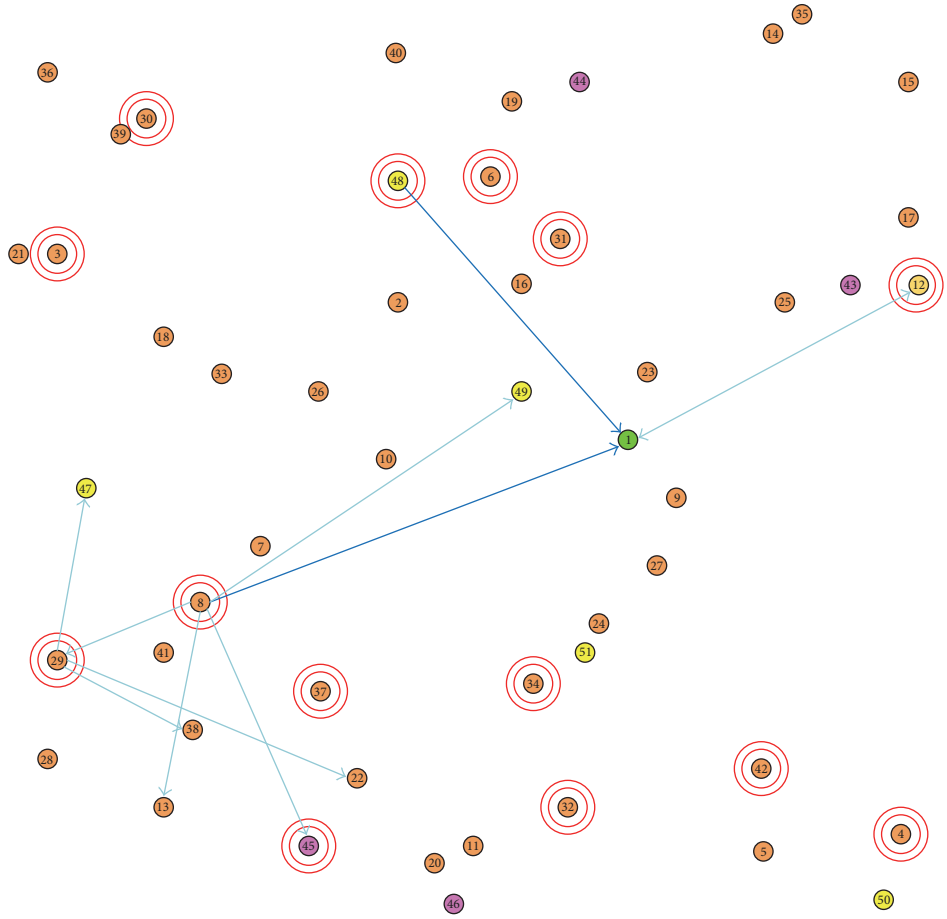

Figure 6: Simulation experiment scenario.

good nodes (orange) and only provide trusted values. Five nodes (pink) are OA and they provide randomly trusted and untrusted data. We also simulated five malfunction nodes (yellow), which always send untrustable data.

Table 3 gives the simulation configuration parameters. We used the Cooja simulator for Contiki 3.0 OS. The simulation ran for two hours on a desktop machine with a Core i7 3.60 GHz processor running Windows 10. The simulation generated 4844 samples of data. Cooja is popular within the WSN and IoT research community and results can be benchmarked with other studies.

3.3. Real Data Validation. To evaluate our method in a real-world scenario, we used 4111 samples of temperature data from February to March 2015 from the city of Aarhus, located in Denmark [17]. This city has a regular temperature range during this time of the year ranging from −3 to 16 degrees Celsius (Figure 7). A total of 500 misbehavior attack samples were simulated using random out-of-range temperature observations (from −23 to 36 degrees Celsius) and injected in the test dataset.

3.4. Classifiers Comparison. To verify the best classifier method to solve this problem we conducted tests with

Table 3: Summary of simulation parameters.

Parameter	Value
Simulator	Cooja under Contiki 3.0 OS
Radio environment	Unit disk graph medium (UDGM): dist. loss
Deployment area	400 m × 400 m
Type & number of nodes	Sky mote, 50 senders & 1 sink
Range of nodes	Trans. range: 50 m, interference range: 50 m
Physical layer	IEEE 802.15.4
MAC layer	IPv6
Network layer	RPL
Transport layer	UDP
Simulation duration	2 h
Sending rate	1 packet in every 20–60 sec

two kinds of classifiers: one-class and multiclass support supervised classifiers. One-class classifiers studied were OneClassSVM, robust covariance (EllipticEnvelope), and isolation forest. Nearest Neighbors (KNeighborsClassifier),

Table 4: Classifiers configuration.

Classifier	Configuration
OneClassSVM	OneClassSVM (cache_size = 200, coef0 = 0.0, degree = 3, gamma = 0.01, kernel = "rbf",, max_iter = −1, nu = 0.01, random_state = None, shrinking = True, tol = 0.001,, verbose = False)
Elliptic Envelope	EllipticEnvelope (assume_centered = False, contamination = 0.1, random_state = None,, store_precision = True, support_fraction = None)
Isolation Forest	IsolationForest (bootstrap = False, contamination = 0.1, max_features = 1.0,, max_samples = "auto", n_estimators = 100, n_jobs = 1, random_state = None,, verbose = 0)
Nearest Neighbors	KNeighborsClassifier (algorithm = "auto", leaf_size = 30, metric = "minkowski",, metric_params = None, n_jobs = 1, n_neighbors = 5, p = 2,, weights = "uniform")
Linear SVM	(C = 0.025, cache_size = 200, class_weight = None, coef0 = 0.0,, decision_function_shape = None, degree = 3, gamma = "auto", kernel = "linear",, max_iter = −1, probability = False, random_state = None, shrinking = True,, tol = 0.001, verbose = False)
Neural Net	MLPClassifier (activation = "relu", alpha = 1, batch_size = "auto", beta_1 = 0.9,, beta_2 = 0.999, early_stopping = False, epsilon = $1e-08$,, hidden_layer_sizes = (100,), learning_rate = "constant",, learning_rate_init = 0.001, max_iter = 200, momentum = 0.9, nesterovs_momentum = True, power_t = 0.5, random_state = None,, shuffle = True, solver = "adam", tol = 0.0001, validation_fraction = 0.1,, verbose = False, warm_start = False)
Naive Bayes	GaussianNB (priors = None)

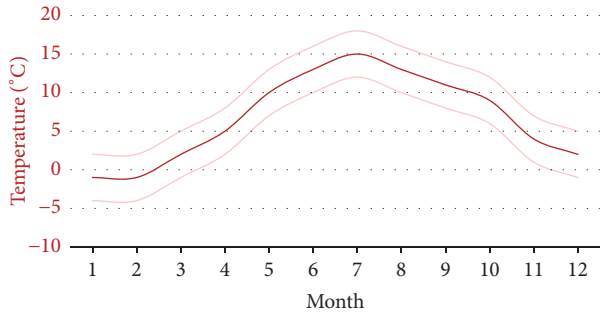

Figure 7: Temperature range observed from the city of Aarhus (extracted from https://en.climate-data.org/location/302/).

linear SVM, Naive Bayes (GaussianNB), and Neural Net (multilayer perceptron) were tested as multiclass classifiers. Table 4 shows the configuration used in each classifier tested.

All classifiers were trained with the same train simulated dataset, with 2000 reads, and tested with a 4844-sample test dataset. A zero-value array was used as a secondary class for multiclass classifiers.

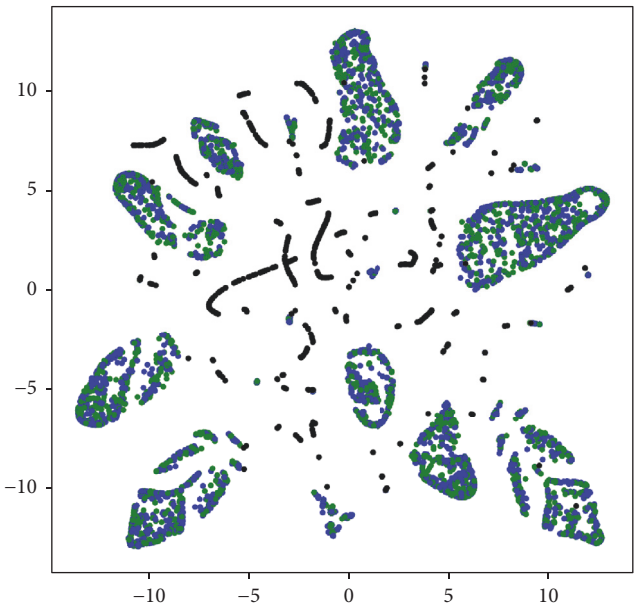

Figure 8: Decision function hyperplane.

4. Results

Our proposed security method was able to detect OA in the IoT with 97% of precision in a real-world dataset and 96% of precision in simulated environment. Compared to other studies, our method was 95% faster and 5% more precise in OAs identification. In a novel way, the elastic window feature helped differentiate broken or malfunctioning nodes among misbehaving devices.

The hyperplane related to the model created by the OneClassSVM classifier is shown in Figure 8. This visualization was generated using the t-SNE technique. The trained dataset (blue points) was grouped to create the decision function boundaries. Meanwhile new normal observations (green points) are likely close to the data used for training. OA (black points) or broken devices are located far from decision function boundaries.

The OneClassSVM classifier efficiently grouped the test data near to the trained dataset, without overfitting. Abnormal samples were presented far from the decision function.

The decision function returns a distance value of an evaluated sample from the decision hyperplane. If a class is identified, normal (trusted) observations have values near 0, while attacks (misbehaving and out-of-range reads) have high distance values (up to −200). In Figure 9 we showed the trust score (distance from decision function) for a set of observed reads.

The two experimental setups were trained with a set of 2000 reads related to the temperature range observed in the

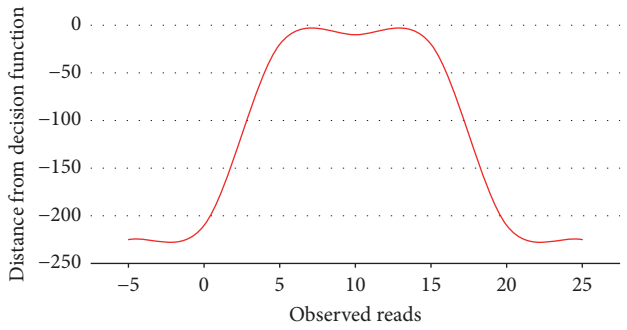

Figure 9: Distance from decision function for observed reads.

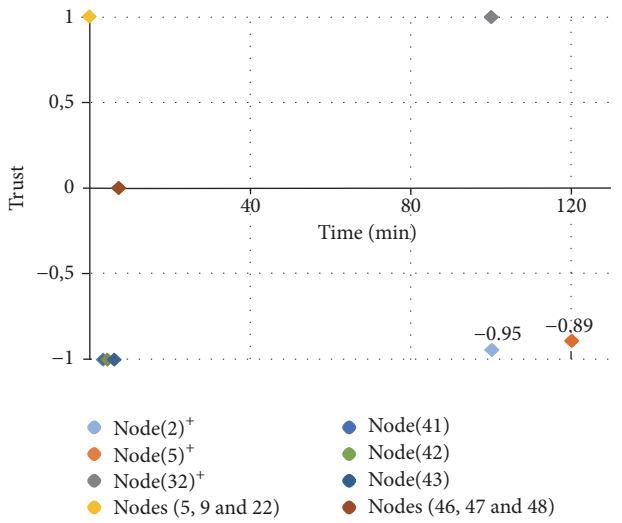

Figure 10: Time spent to identify OA and broken nodes.

city of Aarhus (−3 to 16 degrees Celsius). Test reads in this range obtain decision function distance values near to zero, while other values received low values.

4.1. Simulation Validation. Study [8] utilizing the number, position, and traffic volume of malicious nodes demands approximately 120 min to determine OA. Our method identified OAs in 5 min average time (95% faster) and 96% precision. Figure 10 shows the time spent to identify OA. Nodes 31^* (good), 8^*, and 32^* (attackers) are from compared study [8] and Nodes 5, 9, 22 (goods), 41, 42, and 43 (OA) are from our simulated scenario. Nodes 46, 47, and 48 are broken nodes and also were identified in 7 min. Positive trust scores are related to good nodes and negative trust scores to attackers respectively.

In various situations, not all misbehaving devices are attackers. Some of them may be devices in malfunction status. We did not identify relevant researches in the differentiation IoT OA from errors sent by broken nodes. In this paper, we compare our solution to density-based spatial clustering of applications with noise (DBSCAN) algorithm. DBSCAN identified only two classes. The results in Figure 11 show the true positives (good, attackers, and broken nodes) predicted

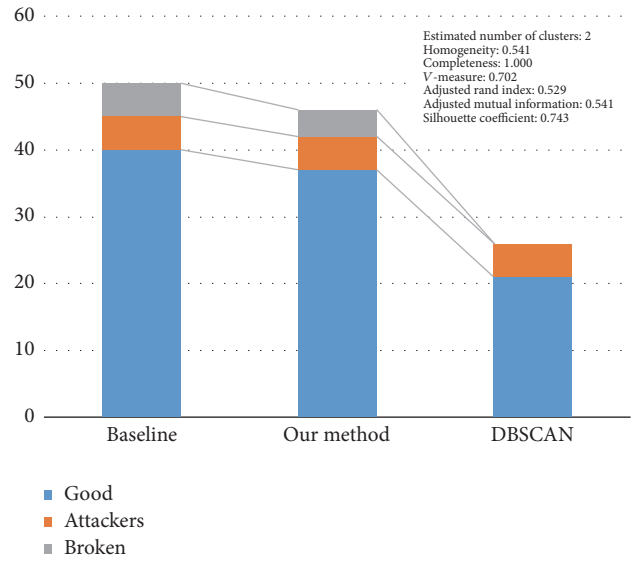

Figure 11: Comparison with DBSCAN clustering algorithm.

Table 5: Comparison with supervised classifiers.

Classifier	Precision	Recall	*F*1-score
Linear SVM	0.88	0.71	0.74
Naive Bayes	0.92	0.82	0.85
Neural Net	0.92	0.81	0.84
Nearest Neighbors	0.91	0.84	0.87
Our method	**0.96**	**0.85**	**0.87**

from the presented method and the DBSCAN, compared to expected nodes (baseline).

The annotated simulated dataset was useful to conduct another validation. We train, test, and compare our method to Nearest Neighbors (KNeighborsClassifier), linear SVM, Naive Bayes (GaussianNB), and Neural Net (multilayer perceptron). The results in Table 5 show our method, with the elastic slide window approach, reaches superior precision, recall, and $f1$ scores.

Our method was able to find three good nodes, two attackers, and two broken nodes more than the supervision methods.

Figure 12 shows how nodes were identified by our method in simulation experiment. The filled points are the actual nodes. The yellow circles are the good nodes, the cyan squares are the attackers, and the red diamonds are the malfunctioning nodes. The mark around the nodes represents the predicted class.

4.2. Real Data Validation. To develop and evaluate the smart middleware, we used 4111 samples of temperature data collected by 115 sensors from February to March 2015 from the city of Aarhus, in Denmark [17]. This city has a regular temperature range during this time of the year ranging from −3 to 16 degrees Celsius. A total of 500 misbehavior samples were simulated using random out-of-range temperature observations (from −10 to 30 degrees Celsius).

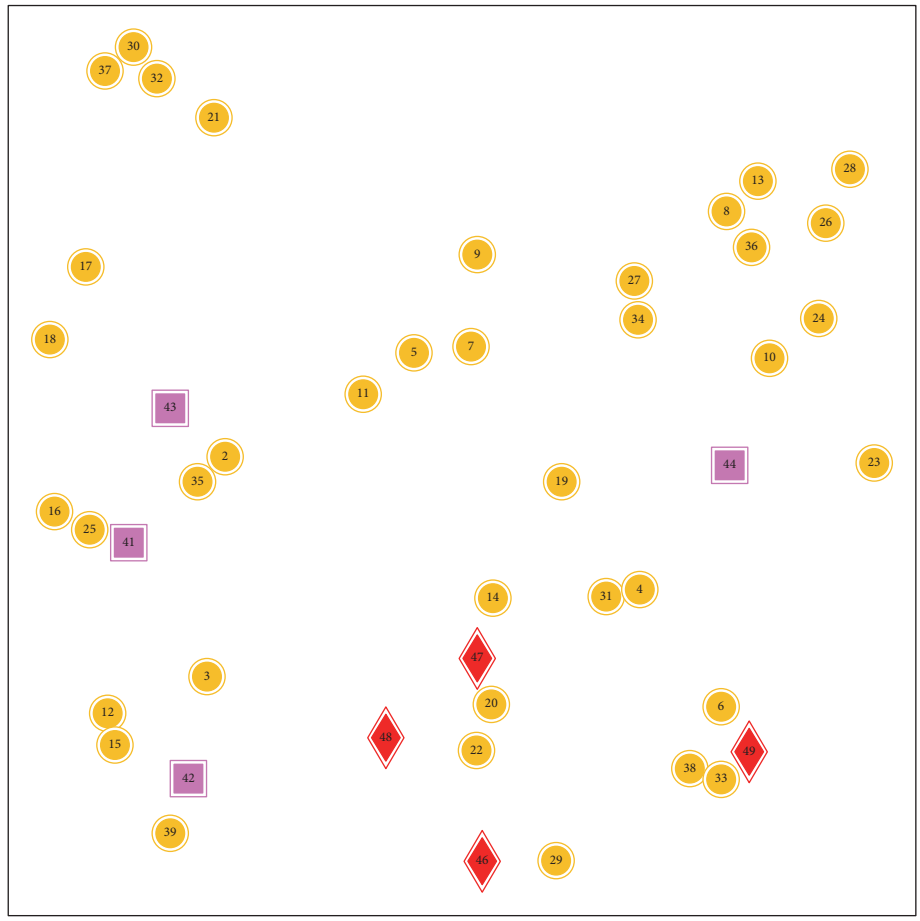

Figure 12: Resources identified by our method.

Similar to the simulation experiment, our method reaches 97% of precision and 95% of recall in classifier goods and misbehaving nodes.

4.3. Classifiers Comparison. The proposed method uses the output results from one-class classifier types. Classifiers like the OneClassSVM classifier are proven to be efficient in outliers detection [18]. We use one-class classifiers because our method only needs to differentiate good (expected) metadata reads from abnormal ones. In real-world implementation, it is hard to train a classifier with each type of attack data. In addition, the system analysis capacity can be enhanced with new one-class classifier for new evaluated metadata.

To identify the best one-class classifier, we compare OneClassSVM, robust covariance, and isolation forest classifiers with the same train and test data (Table 6). The OneClassSVM reaches 99% of precision and 98% of recall in trusted resource prediction.

5. Threats to Validity

This method may have been affected by some validation threats, such as related random simulation outputs, classifiers seeds utilized, and datasets.

Table 6: One-class classifiers comparison for outlier detection in our method.

Classifier	Precision	Recall	*F*1-score
OneClassSVM	**0.99**	**0.98**	**0.99**
Robust covariance	**0.99**	0.90	0.94
Isolation forest	**0.99**	0.90	0.94

Simulations outputs may vary from each run period. To minimize this threat, we ran each simulation three times and use the average results. The scikit-learn library documentation alerts users that it uses random seed values, parameter initialization variable, in the training tasks of the models, which slightly corroborates to different precision results. As in simulations, we annotate average values of three fitting rounds. The real-world dataset used was sanitized removing NaN values and nonrelated data.

The related works session may have been threatened by failing to consider any relevant study. To minimize this risk, we consulted the main indexing databases to verify references and citations to/from selected studies. However, some studies may not have been considered owing to the technical limitation of search engines, the possibility of

electronic databases not representing the complete list of all available studies, and the nonavailability from the Brazilian academic network or access to the printed only version.

6. Conclusion

In this article, we introduced a smart trust management method based on machine learning and an elastic slide window technique that automatically assesses the IoT resource trust by evaluating service provider attributes. Our proposed security method was able to detect OAs in the IoT with 97% of precision in a real-world dataset and 96% of precision in simulated environment. Compared to other studies, our method was 95% faster in OA identification.

The smart trust management method presented here contributes to IoT trust management, protecting systems against OA using less data (starting from two entries of one feature). In a novel way, the elastic slide window feature also helps to differentiate broken or malfunctioning nodes among misbehaving devices.

For future works, we plan to increase the overall precision of our method by using other datasets for training and adjusting the classifier configuration. We also intend to use the elastic slide window to identify other IoT trust related attacks, like opportunistic service attacks, ballot-stuffing attacks, self-promotion attacks, and bad-mouthing attacks.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was partially supported by Fundação de Amparo a Pesquisa de Mato Grosso (FAPEMAT), Brazil.

References

[1] A. Nordrum, "Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated," 2016, Accessed in January 20th 2018, https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated.

[2] I. Yaqoob, E. Ahmed, I. A. T. Hashem et al., "Internet of things architecture: recent advances, taxonomy, requirements, and open challenges," *IEEE Wireless Communications Magazine*, vol. 24, no. 3, pp. 10–16, 2017.

[3] M. A. Ferrag, L. A. Maglaras, H. Janicke, J. Jiang, and L. Shu, "Authentication protocols for internet of things: a comprehensive survey," *Security and Communication Networks*, vol. 2017, Article ID 6562953, 41 pages, 2017.

[4] C. V. L. Mendoza, J. H. Kleinschmidt, R. Chen et al., "Trust management for SOA-based IoT and its application to service composition," *Journal of Machine Learning Research*, vol. 39, pp. 1–6, 2016.

[5] M. Nitti, R. Girau, and L. Atzori, "Trustworthiness management in the social internet of things," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 5, pp. 1253–1266, 2014.

[6] J. Guo, I.-R. Chen, and J. J. P. Tsai, "A survey of trust computation models for service management in internet of things systems," *Computer Communications*, vol. 97, pp. 1–14, 2017.

[7] J.-H. Lee and H. Kim, "Security and privacy challenges in the internet of things [security and privacy matters]," *IEEE Consumer Electronics Magazine*, vol. 6, no. 3, pp. 134–136, 2017.

[8] C. V. L. Mendoza and J. H. Kleinschmidt, "Mitigating on-off attacks in the internet of things using a distributed trust management scheme," *International Journal of Distributed Sensor Networks*, vol. 2015, Article ID 859731, 2015.

[9] H. Hellaoui, A. Bouabdallah, and M. Koudil, "TAS-IoT: Trust-Based Adaptive Security in the IoT," in *Proceedings of the 41st IEEE Conference on Local Computer Networks (LCN '16)*, pp. 599–602, November 2016.

[10] O. Ben Abderrahim, M. H. Elhdhili, and L. Saidane, "TMCoI-SIOT: A trust management system based on communities of interest for the social internet of things," in *Proceedings of the 13th IEEE International Wireless Communications and Mobile Computing Conference (IWCMC '17)*, pp. 747–752, June 2017.

[11] F. Boustanifar and Z. Movahedi, "A trust-based offloading for mobile M2M communications," in *Proceedings of the Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld '16)*, pp. 1139–1143, IEEE, 2016.

[12] W. Li, H. Song, and F. Zeng, "Policy-based secure and trustworthy sensing for internet of things in smart cities," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1-1, 2017.

[13] Y. Yu, Z. Jia, W. Tao, B. Xue, and C. Lee, "An efficient trust evaluation scheme for node behavior detection in the internet of things," *Wireless Personal Communications*, vol. 93, no. 2, pp. 571–587, 2017.

[14] T. Zhang, L. Yan, and Y. Yang, "Trust evaluation method for clustered wireless sensor networks based on cloud model," *Wireless Networks*, pp. 1–21, 2016.

[15] S. M. Sony and S. B. Sasi, "On-Off attack management based on trust," in *Proceedings of the 2016 Online International Conference on Green Engineering and Technologies (IC-GET '16)*, pp. 1–4, 2016.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort et al., "Scikit-learn: machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[17] "C. of Aarhus, Weather Data for the City of Aarhus in Denmark," 2016, Accessed in January 20th 2018, http://iot.ee.surrey.ac.uk:8080/datasets.html.

[18] W. Khreich, B. Khosravifar, A. Hamou-Lhadj, and C. Talhi, "An anomaly detection system based on variable N-gram features and one-class SVM," *Information and Software Technology*, vol. 91, pp. 186–197, 2017.