# An edge-based smart network monitoring system for the Internet of Vehicles

Naercio Magaia
*COPELABS, Universidade Lusófona*
Lisbon, Portugal
*School of Engineering and Informatics*
*University of Sussex*
Brighton, UK
0000-0002-6613-1666

Pedro Ferreira
*Instituto Superior Técnico*
*Universidade de Lisboa*
Lisbon, Portugal
pedro.m.a.ferreira@tecnico.ulisboa.pt

Paulo Rogério Pereira
*INESC-ID/INOV*
*Instituto Superior Técnico*
*Universidade de Lisboa*
Lisbon, Portugal
prbp@inesc.pt

***Abstract*—The Internet of Vehicles (IoV) is the future of transportation. It will be present everywhere and will have a huge impact on our lives. However, there are plenty of aspects to consider while studying these networks, such as data dissemination, cybersecurity threats and vulnerabilities. For an IoV to work efficiently, data needs to spread through it efficiently. However, the dynamics of vehicular environments due to frequent node mobility and nodes' misbehavior poses many challenges to efficient data dissemination. Therefore, a deep learning-based monitoring system that is capable of detecting anomalies in the network and identifying known misbehavior is proposed. Performance evaluation shows that the monitoring system can identify well-known attacks with a very high success rate. Besides, the algorithm is also capable of detecting other types of misbehavior without labeling them.**

***Index Terms*—Internet of Vehicles, Network Monitoring, Deep Learning, Edge**

## I. Introduction

The Internet of Vehicles (IoV) is becoming the next transformation in the world of transportation. Its main goal is safety, comfort, and prompt delivery of the vehicles' occupants with minimum impact on the environment [1]. With this goal in mind, there are several applications for this technology, such as management of network traffic, reduction of traffic jams, alert users about any hazard, and call for specific help and send information about the victims in case of an accident [2].

These goals can only be achieved through communication among IoV objects (i.e., vehicles, pedestrians, Road Side Units - RSUs) and public networks. However, for the IoV to work at its full potential, a huge amount of data has to be able to spread throughout the network. The dissemination of this data leads to numerous cybersecurity threats and vulnerabilities, which can lead to data breaches where the attacker gathers data from other nodes to perform threats that can impact the environment itself. To detect these attackers, a monitoring system, e.g., Intrusion Detection System (IDS), a system to detect any anomaly or intrusion, can be deployed on the network. Due to the amount of data and possible vulnerabilities most networks or systems are exposed to, many monitoring systems use nowadays Machine Learning (ML) techniques.

Most of the IoV challenges have also appeared in other fields of study, such as the Internet of Things (IoT), given the similarities between these two areas. The same architecture schema is being used to model both IoV and IoT, splitting them into three layers: Vehicles (Things), Edge, and Cloud [3]. The Vehicles layer is mainly responsible for data collection and actuation to control the physical world. Most devices in this layer are resource-constrained in terms of computational power, storage, and energy. The Edge layer is introduced to help end devices. First, computation-intensive tasks can be offloaded to edge devices. Second, the edge layer can mask communication heterogeneity among end devices and connect them to the Internet. Third, edge devices help manage end devices. At last, the Cloud layer is utilized to store, process, and analyze the collected data and provide the additional support needed by many applications [4].

A monitoring system needs to be flexible enough to work everywhere and adapt to every city, village, or any other place it might be deployed. It has to learn to accomplish all goals it was designed to achieve. By considering the above, and due to the generated amount of data, an ML-based monitoring system is desirable. However, such systems may require considerable computational power and a short response time. To comply with such requirements, one could use the computational and storage power closer to the end-users, i.e., at the edge of the IoV network.

Nonetheless, the following aspects should also be considered: (i) what if the algorithm stops working? (ii) what if the algorithm does not share with the vehicles the correct data?, and (iii) what if some nodes act in a way that may harm the others? These problems may occur if the network is attacked.

Aiming to address the latter, in this paper, we propose a Deep Learning (DL) based monitoring system at the edge layer to detect any anomaly and classify node behaviors in the network, helping mitigate the impact of misbehaving nodes.

The remainder of this paper is structured as follows: Section

This work was supported by H2020-MSCA-RISE under grant No 101006411, by Portuguese national funds through FCT, Fundação para a Ciência e a Tecnologia, under project UIDB/50021/2020 and UIDB/04111/2020, and by Portuguese national funds through FITEC-Programa Interface, with reference CIT "INOV-INESC Inovação-Financiamento Base".

II presents related work. Section III presents system and misbehavior models. In Section IV, the smart monitoring system is presented. Section V presents performance evaluation. Finally, Section VI presents concluding remarks.

## II. Related Work

Even with the protective measures implemented by the different security mechanisms and protocols, attackers are one or two steps ahead in exploiting vulnerabilities, which highlights the constant need for optimized solutions.

IDSs appeared in conventional computer networks to give an additional layer of security to the strategies employed by network administrators. Utilizing IDSs as a security tool enables to constantly analyze network packets, verifying whether they correspond to previously defined attack signatures [5].

Vehicular environments' characteristics, such as different types of communications, high mobility, and node density, pose challenges for the utilization of IDSs. The choice of IDSs as security tools presents two distinct learning opportunities for network administrators: (i) the protection of the messages exchanged between different entities (i.e., vehicles, RSUs, among others) that use Vehicle-to-Everything (V2X) communications, besides identifying the occurrence of attack attempts, and (ii) in dynamic IoV scenarios, checking which vehicle service has the highest attacks incidence is an important feedback for future decision-making on how to improve security measures as well as learning the methodology for carrying out attacks.

Kumar and Chilamkurti [6] have used a Learning Automata (LA) to capture vehicle characteristics and the Markov Chain Model (MCM) to represent vehicle states. The authors proposed an IDS to identify any anomalies that may occur in the network based on a parameter called Collaborative Trust Index (CTI). Aloqaily et al. [7] proposed the identification of Probing, User to Root, Remote to User, and denial-of-service attacks (DoS) in electric vehicular networks for Smart cities using an IDS. Liang et al. [8] proposed an IDS for the identification of False Information and Sybil attacks. The detection algorithm consists of a neural network called GHSOM.

Malhi and Batra [9] developed a generic framework for the identification of anomalies in vehicular networks. The threats identified in their study were distributed DoS (DDoS) and masquerade attacks, which cause network unavailability and send false information to the "victim vehicles", respectively.

Loukas et al. [10] used an automated car to test an IDS developed with DL algorithms. To solve the performance capacity problem of the automated car, cloud computing was used to process the collected data.

On-road messages' exchange increases management and classification complexity, as the network infrastructure grows to make new services available. On the other hand, the creation of new services or their increase can mean the appearance of new security breaches, requiring the development of practical security tools to protect the network and vehicles [11]. Data capture and inaccessibility of information are problems that extend to systems connected to the Internet. Some solutions using ML/DL algorithms for threat identification (e.g., botnets, DDoS, black hole, and spoofing attack) are listed in the following.

Garip et al. [12] proposed an adaptive detection mechanism, called SHIELDNET, to identify botnets on the network. For the identification of DDoS attacks (i.e., brute force, TCP-SYN, UDP, and HTTP floods), Nie et al. [13] used Convolutional Neural Network (CNN) for the development of a data-driven IDS, i.e., a hybrid IDS that is signature-based and anomaly-based, to identify the DDoS attack in the exchange of messages between On-Board Units (OBU) and RSUs. Alheeti et al. [14] proposed an IDS that uses Artificial Neural Networks (ANNs) and fuzzified data to identify and correct the black-hole attack problem in vehicles with the auto-driving system. Kosmanos et al. [15] developed an IDS for the identification of spoofing attacks in electric vehicles. In addition to using ML, it also makes use of Position Verification using Relative Speed (PVRS) to optimize the results obtained.

Most state-of-the-art solutions were proposed for traditional vehicular networks, i.e., VANETs, hence not being suitable for IoV. Our proposed approach is generic in the sense that if enough training data is provided, it can also detect DoS, DDoS, black hole attacks, among others. In addition, the huge amount of data generated by IoV objects brings new challenges. ML-based solutions become more computationally demanding, hence the need for computation at the edge of the network. Such challenges require constant search for optimized solutions, easy configuration, and adaptation to ensure that human lives are not at risk due to different security problems.

## III. Preliminaries

### A. System Model

The proposed IoV network is composed of three layers, namely Vehicles, Edge, and Cloud. There are mainly two groups of nodes: mobile and stationary. On the first group, at the Vehicle layer, there are cars and buses. On the second group, at the Edge layer, there are the RSUs.

Mobile nodes can offload data to the nearest RSU to be processed. Once the computation is finished, the output will be sent through other nodes in the direction of the desired node. Nevertheless, RSUs are all connected through the Edge, and, thus, some data can be shared between them.

### B. Misbehavior Model

There are plenty of behaviors or malfunctions that may have a huge impact on the network [16]. Therefore, the following behaviors were implemented. The *identity (id) spoofing* attack corresponds to a node that successfully identifies itself as another one or tries to have a different role in the network. The *sybil attack* occurs when a node, i.e., the attacker, subverts the reputation system by creating a large number of pseudonymous identities and uses them to gain influence in the network. The *node not working* behavior consists of a node dropping or simply not sending any of the messages it receives. In

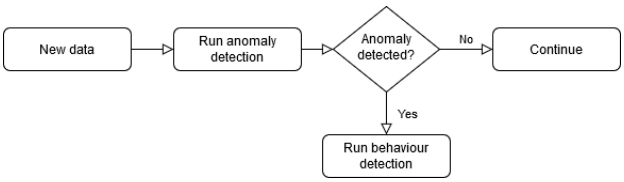


Fig. 1. The monitoring flowchart

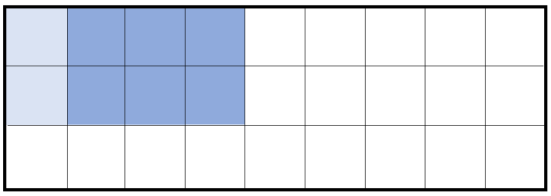

Fig. 2. An example of the kernel(3x2) and stride(1) on a 9x3 array

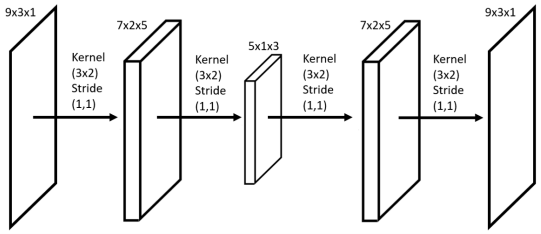


Fig. 3. Architecture of the Convolution Auto Encoder

addition, the node may have a malfunction on its 802.11p communication interface. Therefore, it can only communicate with RSUs. These actions were implemented on the three different types of nodes, namely RSUs, cluster heads, and normal nodes.

## IV. The Smart Monitoring System

This section describes two DL approaches, namely misbehavior (or anomaly) detection and misbehavior identification, which run at the Edge layer. The first algorithm uses data from each node and infers its misbehavior probability. If the output is high, it will indicate the most probable behavior, as shown in the flowchart of Figure 1.

It is important to mention that the objective of this work is only to detect and, if possible, to identify the malicious behavior. Its aims is to help the network manager by providing alerts. It is not intended to affect any node or the network itself directly.

### *A. Misbehavior Detection*

The data provided to the algorithm are the: (i) type of node (e.g., normal node, cluster head, or RSU), (ii) ego in the network, (iii) number of nodes in range, (iv) number of messages received (v) number of messages sent, (vi) buffer size, (vii) node location, and (viii) social strength with its three strongest connections. Please note that the location is provided as Cartesian coordinates (X, Y).

This data is stored in blocks of three arrays that form a 2D array that contains the data of a given node at three different times. The number of samples in each block of data is limited to three to consider the node's past without having a huge impact on the algorithm's performance. Therefore, the input format is $9 \times 3$.

All data used is sent by the vehicles to the Edge every time they communicate with an RSU. By not sharing their data, it means that they did not interact with the Edge. Consequently, their role in the network will be less important, and the impact of their actions will be lower.

This data enters a convolutional autoencoder [17], [18] that uses the convolution property to simulate data dependency over time. The algorithm aims to encode the input data into a smaller space than the original and then, from the encoded data, decode it to restore the original input. The performance of the algorithm is measured by the mean squared error between the input and the output data. Therefore, the goal of the algorithm is performing a robust data analysis, thus, building a robust anomaly detector.

The autoencoder used is formed by convolutional layers. These layers receive a 3D input and convolve it with a set of kernels, or filters, and apply an activation function to the filter outputs, which in this case is a Rectified Linear Unit [19]. Each kernel has localized support in the first two spatial coordinates and a full range on the depth on the input (third coordinate). It will compute the value of each neuron of the next layer according to this paradigm.

All layers of the algorithm have the same parameters, a kernel of $3 \times 2$ and a stride of 1. The kernel can be seen as a filter, where its size is the filter size. The amount of the filter shift is given by the stride.

Figure 2 presents an example of an array of $9 \times 3 \times 1$ neurons with a kernel and stride with the same parameters as used in the algorithm after the first iteration. The blue represents the area covered by the kernel, and the lighter blue the area that was already covered.

At each iteration, the kernel analyzes the data it is covering and computes a single value to represent it. In the end, the data is reshaped according to Eq. 1, which has to be applied to both dimensions.

$$output\_vol = \frac{(input\_vol - kernel\_size)}{stride} + 1 \qquad (1)$$

The encoder consists of two convolution layers that transform the input data into five neuron arrays of $7 \times 2$ and then three arrays of $5 \times 1$. Then, the decoder is two layers of the reverse function with the same parameters, as shown in Figure 3.

To conclude, in the algorithm used (see Figure 3), each neuron of the second layer is computed based on the first layer neurons covered by a kernel ($3 \times 2$) that shifts one position (i.e., a stride) every time it is applied. In this case, the depth of the first layer is just one. Therefore, the kernel only considers six neurons each time. However, for the other layers, as their

depth is higher than one, the kernel covers more neurons. For example, at each iteration between layer 2 and 3, 30 (i.e., $3 \times 2 \times 5$) neurons are considered.

A library called DeepLearning for Java [20] is used to implement this neural network. It is trained in a different program (please refer to Section V-B for more information). Only the testing part runs at the same time as the simulations. As mentioned before, the algorithm issues a score that is the mean squared error between the input and the output, and values closer to zero represent nodes whose behavior is closer to the normal. When this score is higher than a threshold, the node is labeled as misbehaving. This threshold is adjusted during the simulation to be adapted to each case. Its initial value is the highest score obtained while testing the algorithm. The mean score of this test as its relation with the highest score will also be stored and will be used as a guideline to the update of the threshold. During the simulation, every time the algorithm runs, the $mid\_value$ is adjusted, as shown in equation 2.

$$avg\_score = avg\_score * 0.9 + new\_score * 0.1 \quad (2)$$

Then, the relation obtained during testing is applied to this average and the threshold is afterwards updated.

### B. Misbehavior Identification

Besides misbehavior detection, the Edge should also be able to label misbehavior actions. Therefore, when the previous algorithm raises a flag, the data is forwarded to a new neural network.

This DL algorithm has the same input as the previous one but is now a classification problem. It has only five different outputs: Sybil attack, node not working, identity spoofing, normal behavior, and new unknown behavior.

It starts with a convolution layer to simulate the dependency of data over time and consists of three dense layers and a softmax function to discern the labels. The choice of the number of layers and the number of neurons is a balance between the flexibility of the network, as with more layers, the network can synthesize a wider variety of nonlinear functions with fewer neurons, and the difficulty to train given that a deeper (i.e., more layers) and denser (i.e., more neurons) network requires more computation (see Figure 4).

The data set used to train this network is divided into the training and the validation data sets. This model uses the validation set to implement the early stopping, i.e., a technique used to ensure that the network is not over-fitted to the trained data. Another aspect to be taken into account is the fact that the network cannot be trained to detect new unknown behaviors, as that would be a paradox. Therefore, the neural network is trained to label the four known behaviors, and when the confidence in the result is low, the behavior is classified as an unknown one. The output of this algorithm is an array with four positions, one for each label that represents how confident is the network in assigning the label to the data set. The sum of the four positions of the array is always one. Therefore, given a data set, if the confidence is not higher than $0.5$ for any label, the data can be labeled as unknown behavior.

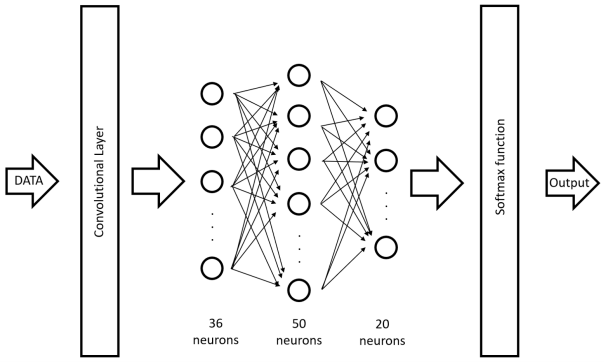


Fig. 4. Behavior identification neural network

## V. Performance Evaluation

### A. Simulation Model

The proposed IoV network was implemented on the ONE simulator [21]. It is assumed that all nodes are equipped with a wireless interface using an 802.11p Wireless Access in Vehicular Environments (WAVE) [11] with a transmission range of 100 m and a transmission rate of 10 Mbps, to communicate with each other. In addition, each vehicle is also equipped with a more powerful wireless interface that can only be used to communicate with RSUs. This interface has a transmission range of 250 m but with the same transmission rate.

The Helsinki workday (HW) scenario consisted of 56 nodes divided into five groups of home, office, and meeting spots following the working day movement (WDM) model . WDM simulates the usual movement of a person spending the night at home, then going to work for eight hours and after that, there are a few meeting points replicating stores, cinemas, restaurants, and other places where people may go after work. There are also two buses for each of the eight routes available in the scenario. Cars' buffers vary from 64 to 256 MB; meanwhile, all buses have 256 MB. In addition, there are also 30 RSUs.

### B. Training data

All data used to train this neural network has been collected during several simulations with different parameters such as the number of nodes, movement models, or message size, to build a model as flexible and accurate as possible.

The simulations were performed by dividing the nodes into three groups: RSUs, buses, and vehicles. Considering that the movement of the buses and the position of the RSU are always the same, only the movement pattern of the vehicles was changed.

The simulator used has data concerning typical home, office, and meeting locations in Helsinki, divided into eight groups. In the majority of the simulations, the WDM movement model was used as it simulates daily routines.

During the simulation, RSUs would write their data into a file every hour, and the data concerning the other nodes was written when they encountered RSUs.

### C. Results

Network monitoring consists of two different algorithms: (i) one that detects anomalies and (ii) the other that tries to identify them.

This evaluation consisted of several simulation runs where some nodes were programmed to misbehave and the Edge tried to detect them. Every time the first algorithm detected an anomaly, the data also passed through the second one to classify the behavior.

The following metrics were considered for evaluating the algorithms: true positive, false positive, true negative, and false negative. A true positive is when an attacker is correctly labeled. A false positive is when a normal behavior is classified as misbehaving. A true negative is when a normal node is labeled correctly. A false negative is when an attacker is mislabeled.

Before using the smart monitoring system on a real simulation, the algorithms were trained and tested. Their training consisted of uploading a data set and the algorithms updated their weights to best perform on the given data. The training data was obtained from several simulations runs with different parameters. The misbehavior detection took approximately 6 minutes to train while the misbehaviour identification took 2 minutes.

After the training, the algorithms were tested with a set of already known data (independent from the training set). The first algorithm was trained to extract the main features of a data set, and based on those, replicate the input data. During its evaluation, the algorithm was capable of reproducing the input data with and, in the worst case, only a mean squared error of 0.2. The second algorithm was tested with 150 data samples.

In the following, attackers behave maliciously from the beginning until the end of the simulation. Their data is monitored every time they contact an RSU, and the algorithm does not consider any prior label given to nodes. Instead of testing a node based on all its history, the tests only take into consideration the data sample acquired at each moment. Therefore, they are presented below for different attack situations. Usually, each simulation has five misbehaving nodes, and during the simulation time, each node is inspected by the algorithm 20 to 30 times, depending on its movement patterns.

*1) HW scenario:* First, the misbehavior detection algorithm is used to monitor a network where all nodes behaved normally. In this case, 1500 tests were performed over 106 nodes. From these tests, 117 were considered as a potential anomaly. However, the latter, the second algorithm labeled 108 as normal behavior. This indicates that from 1500 samples, only 9 were false positives, i.e., only $0.6\%$.

Considering that each node's attack signature, i.e., each node's behavior, is the same at the beginning of the simulation, if the algorithm is applied, all nodes would have the same

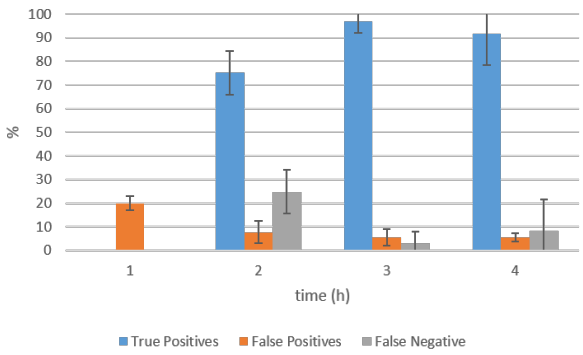


Fig. 5. Evolution of classification metrics for the id spoofing attack

classification. Over the simulation, the signatures become more and more distinct. Therefore, in the first instance, the time that the systems need to warm up, i.e., the difference between the signature of a normal and an attacker node, is studied. With this objective, several simulation runs focusing on the first hours of the simulation were performed for the first two behaviors (i.e., Sybil and id spoofing) with distinct parameters. Here, the evolution of the true positives, false positives, and false negatives were analyzed and once the values of the true positives were higher than the values of false negatives and false positives, it indicated that the system has warmed up and the algorithm were ready to run. For instance, Figure 5 presents the metrics' evolution for the id spoofing attack.

During these tests, the classification metrics started to stabilize after 3 hours. Therefore, 10800 seconds is chosen as warm up time.

*a) Sybil Attack:* For the Sybil attack, several simulations were performed varying the attacker movement model seed and the number of replicas it creates. Among all these runs, the misbehavior detection algorithm analyzed 597 attack situations, and from these 407 were labeled as having an anomaly and sent to the misbehavior identification algorithm that labeled all 407 as Sybil attacks. This means that once the anomaly detector flags an attacker, the misbehavior identifier labels this attack correctly. In addition, this false negative rate (approx. $32\%$) derives from the similarity of an attacker with a node with an important role in the network, i.e., a node with high ego betweenness centrality. Moreover, during this study, a total of 8000 nodes were monitored, and only 16 were mislabeled as Sybil attackers, which leads to a false positive rate of only $0.22\%$.

During this analysis, it was concluded that when the attackers create more replicas, it is easier to detect them. The false negatives are significantly higher when testing with three replicas than with the most used configuration, with five replicas.

*b) Identity Spoofing:* Regarding the id spoofing attack, when a node identified as an RSU contacts the Edge and is not registered as an RSU, the monitoring system already knows that something abnormal is occurring. Therefore, it will always

run the misbehavior identifier.

During the testing period, the algorithm runs the data of 18000 tests, from which 389 were attack situations, and 370 were labeled as attackers. In addition, during these tests, there were 100 false positives, i.e., nodes with normal behaviors being labeled as attackers.

*c) Node Not Working:* The difference between a node not working and a node whose role in the network is not important is small. A node that at a given time stopped working and a node that went to an isolated area and cannot connect to any one, result in a similar behavior. This behavior was not tested on RSUs, hence if one of these nodes was not working, it would not communicate with the Edge. Therefore, it is assumed that the Edge is able to detect this anomaly without the algorithm.

Regarding the results obtained, the true positive rate was only around $50\%$ for the node not working behavior. As mentioned before, this behavior has a signature similar to the normal behavior and even when the anomaly detector captures these nodes, it is not guaranteed that the behavior identifier will not label it as a normal node with a normal behavior. To have a greater certainty about this behavior, it is advisable to wait for at least one more iteration on the algorithm and check if the classification stays the same. On the other hand, the false positive rate on this behavior, just like on the others, is close to zero. In a total of 5000 evaluations, only 33 were misclassified as a node not working.

## VI. Conclusions and Future Work

This article proposes a DL-based monitoring system that is capable of detecting anomalies in the network and identifying known misbehavior. This system takes advantage of the Edge layer and the location of the RSUs to comply with the IoV paradigm. It consists of two different DL algorithms, namely the misbehavior detection, which detects anomalies in the network, and misbehavior identification, identifying these anomalies.

In the HW scenario, which was used for training, the monitoring system was able to detect and identify $68\%$ of the Sybil attacks, $95\%$ of the id spoofing attacks, and $52\%$ of the nodes not working. This result obtained for the nodes not working was because this behavior was similar to that of a node in an isolated area. The misbehavior identification algorithm was able to correctly classify all the Sybil attackers that were sent by the misbehavior detection algorithm.

Continuously training the monitoring system with live data aiming to adapt it better to the network it is monitoring is left for future work. Saving nodes' classifications aiming to analyze the evolution of their behavior over time is also left for future work. The latter will increase the certainty of a given classification by comparing it with the previous ones.

## References


[1] N. Magaia, G. Mastorakis, C. Mavromoustakis, E. Pallis, and E. K. Markakis, Eds., *Intelligent Technologies for Internet of Vehicles*, ser. Internet of Things. Cham: Springer International Publishing, 2021.

[2] L. Silva, N. Magaia, B. Sousa, A. Kobusińska, A. Casimiro, C. X. Mavromoustakis, G. Mastorakis, and V. H. C. de Albuquerque, "Computing paradigms in emerging vehicular environments: A review," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 3, pp. 491–511, 2021.

[3] N. Magaia, P. Gomes, L. Silva, B. Sousa, C. X. Mavromoustakis, and G. Mastorakis, "Development of mobile iot solutions: approaches, architectures, and methodologies," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[4] K. Sha, R. Errabelly, W. Wei, T. A. Yang, and Z. Wang, "Edgesec: Design of an edge layer security service to enhance iot security," *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, pp. 81–88, 2017.

[5] T. G. Nguyen, T. V. Phan, B. T. Nguyen, C. So-In, Z. A. Baig, and S. Sanguanpong, "SeArch: A Collaborative and Intelligent NIDS Architecture for SDN-Based Cloud IoT Networks," *IEEE Access*, vol. 7, pp. 107 678–107 694, 2019.

[6] N. Kumar and N. Chilamkurti, "Collaborative trust aware intelligent intrusion detection in VANETs," *Computers and Electrical Engineering*, vol. 40, no. 6, pp. 1981–1996, aug 2014.

[7] M. Aloqaily, S. Otoum, I. A. Ridhawi, and Y. Jararweh, "An intrusion detection system for connected vehicles in smart cities," *Ad Hoc Networks*, vol. 90, p. 101842, 2019.

[8] J. Liang, J. Chen, Y. Zhu, and R. Yu, "A novel Intrusion Detection System for Vehicular Ad Hoc Networks (VANETs) based on differences of traffic flow and position," *Applied Soft Computing Journal*, vol. 75, pp. 712–727, feb 2019.

[9] A. K. Malhi and S. Batra, "Genetic-based framework for prevention of masquerade and DDoS attacks in vehicular ad-hocnetworks," *Security and Communication Networks*, vol. 9, no. 15, pp. 2612–2626, oct 2016.

[10] G. Loukas, T. Vuong, R. Heartfield, G. Sakellari, Y. Yoon, and D. Gan, "Cloud-Based Cyber-Physical Intrusion Detection for Vehicles Using Deep Learning," *IEEE Access*, vol. 6, pp. 3491–3508, dec 2017.

[11] N. Magaia and Z. Sheng, "ReFIoV: A novel reputation framework for information-centric vehicular applications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1810–1823, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8574942/

[12] M. T. Garip, J. Lin, P. Reiher, and M. Gerla, "SHIELDNET: An Adaptive Detection Mechanism against Vehicular Botnets in VANETs," in *IEEE Vehicular Networking Conference, VNC*, vol. 2019-Decem. IEEE Computer Society, dec 2019.

[13] L. Nie, Z. Ning, X. Wang, X. Hu, J. Cheng, and Y. Li, "Data-Driven Intrusion Detection for Intelligent Internet of Vehicles: A Deep Convolutional Neural Network-Based Method," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2219–2230, apr 2020.

[14] K. M. Alheeti, A. Gruebler, and K. D. McDonald-Maier, "An Intrusion Detection System against Black Hole Attacks on the Communication Network of Self-Driving Cars," in *Proceedings - 2015 6th International Conference on Emerging Security Technologies, EST 2015*. Institute of Electrical and Electronics Engineers Inc., mar 2016, pp. 86–91.

[15] D. Kosmanos, A. Pappas, L. Maglaras, S. Moschoyiannis, F. J. Aparicio-Navarro, A. Argyriou, and H. Janicke, "A novel Intrusion Detection System against spoofing attacks in connected Electric Vehicles," *Array*, vol. 5, p. 100013, mar 2020.

[16] N. Magaia, P. R. Pereira, and M. P. Correia, *Cyber Physical Systems: From Theory to Practice*. CRC Press, 2015, ch. Security in Delay-Tolerant Mobile Cyber-Physical Applications.

[17] A. Ng *et al.*, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

[18] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6.

[19] K. Hara, D. Saito, and H. Shouno, "Analysis of function of rectified linear unit used in deep learning," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–8.

[20] E. Deeplearning4j, "Deep Learning for Java," 2021. [Online]. Available: https://deeplearning4j.org/

[21] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. New York, NY, USA: ICST, 2009.