Deep Learning for In-Vehicle Intrusion Detection System

Elies Gherbi[1,2(✉)], Blaise Hanczar[2], Jean-Christophe Janodet[2], and Witold Klaudel[1]

[1] Institute of research systemX, 91120 Palaiseau, France
elies.gherbi@systemX.fr, {elies.gherbi,witold.klaudel}@irt-systemX.fr
[2] IBISC, Univ Evry, Université Paris-Saclay, Saint-Aubin, France
{blaise.hanczar,jeanchristophe.janodet}@univ-evry.fr

Abstract. Modern and future vehicles are complex cyber-physical systems. The connection to their outside environment raises many security problems that impact our safety directly. In this work, we propose a Deep CAN intrusion detection system framework. We introduce a multivariate time series representation for asynchronous CAN data which enhances the temporal modelling of deep learning architectures for anomaly detection. We study different deep learning tasks (supervised/unsupervised) and compare several architectures, in order to design an in-vehicle intrusion detection system that fits in-vehicle computational constraints. We conduct experiments with many types of attacks on an in-vehicle CAN using SynCAn Dataset.

Keywords: Intrusion detection system · In-vehicle security · Deep learning · Anomaly detection · Time series

1 Introduction

Future applications like autonomous transportation require various technologies that allow the vehicles to interact with other vehicles (VANETs), pedestrians and road infrastructure. These controllers make the vehicles more and more connected with the external world, which allow new functionalities but also dramatically increase the security risk.

In this work, we focus on the CAN bus, which is *de facto* standard for in-vehicle communication. In-vehicle networks technologies must ensure a set of requirements, some of which are time-critical and safety-related. CAN protocol uses broadcast communication techniques. Each node in the network can send and receive a packet to/from the bus [1]. CAN bus contains several vulnerabilities. It does not include the different security criteria in its design. It lacks security facilities like message authentication, that prevents an unauthorized node from joining the communication and broadcast malicious messages to other nodes. It also lacks encryption because it would make overhead for real-time communication. These weaknesses of the protocol are as many possibilities left to hackers to attack, as shown in the cyber-security literature [1,14]. Several

H. Yang et al. (Eds.): ICONIP 2020, CCIS 1332, pp. 50–58, 2020.
https://doi.org/10.1007/978-3-030-63820-7_6

attacks scenarios have been demonstrated on vehicles. *E.g.*, [12] has performed four different tests on the control window lift, warning light, airbag control system and central gateway.

We focus on anomaly detection based intrusion detection using the advances in deep-learning architectures to handle the CAN data structure. To do so, we define three levels for the in-vehicle IDS framework: 1) CAN data level, 2) sequence modelling level and 3) detection level.

This research aims to propose a general in-vehicle IDS using deep learning; we propose a representation for CAN data, and experiments several deep learning architectures for sequence modelling. We finally get an anomaly detection IDS that meets both the needs and the constraints of in-vehicles systems.

2 Background and Related Works

CAN Data. Any CAN message is composed of several fields: an ID and a 64-bit payload. The ID is unique and defines the content (set of signals encoded in a range of several bits) and the priority of the message [6]. A timestamp is added whenever the message is captured. From those characteristics, many representation and feature can be derived as time interval, sequence, time-frequency. Notice that the payload signals are encoded, so it is hard to obtain the signals values without the constructor specifications. CAN Intrusion Detection Systems generally intervene at two levels: either the flow level or the payload level. In flow-based approaches, a flow is a group of packets sharing common properties during a specific period. In payload-based approaches, the payload represents the information carried by the packet. This information is exchanged between the ECUs, and their interpretation reflects the behaviour of the vehicle.Notice that some attacks may highly impact the communication flow, like flooding attack, and will be more easily detected at the flow level. On the other hand, other types of attacks are not visible in the communication flow and be detected only at the payload level. To the best of our knowledge, the actual IDS are either flow based or payload based, while our proposition is based on both.

CAN Intrusion Detection System. We here focus on anomaly-based approaches only. The method starts by defining a model (*profile*) that specifies the actual normal behaviour of the system. Any behaviour that does not conform to the normal profile is considered as an anomaly. The anomaly-based IDS have many advantages: there is no need to maintain a database of signatures, and they can detect unknown attacks since at least from a theoretical standpoint, each attack compromises the normal behaviour of a system.

With the advances of deep learning for time series, many deep learning architectures have been used to solve sequential modelling problems, and anomaly detection based CAN IDS is one of them. In [19], the authors propose a deep convolutional network classifier IDS, a supervised approach designed to learn about traffic patterns and detect different malicious behaviours. They reduce the unnecessary complexity in the architecture of inception-resnet. They have

tested their model on different types of attacks using bit-level CAN dataset where each input consists in 29 sequential CAN IDS. In [8], the authors tackle a large dataset with an extensive type of attacks (SynCAN dataset). They propose a deep learning architecture that handles the structure of CAN data; It is composed of independent recurrent neural networks, one for each ID in the input. The goal of those separated LSTM is to learn about the state of each ID. The whole state of the network is represented by a joint vector of the outputs of all separated LSTM. The second part of the architecture takes the joint latent vector as an input for an autoencoder (fully connected network) that enables unsupervised learning; The task of the autoencoder is to reconstruct the signals for each possible input message based on the joint vector. The attack detection occurs by observing the deviation between the signal value of a message at the current time step with its reconstruction. [16] proposes an IDS by analyzing bit-level CAN message, using LSTM to predict the next message based on the history size of 10 messages; If the distance between the predicted message and the actual message is bigger than a threshold, then the message is an anomaly.

We note that, in the literature, many dimensions can be considered to design the CAN IDS. The used data highly impacts the type of detectable attacks, as well as the design of the model that must learn about a broad range of situations to ensure that the model encompasses the exhaustive space of normality, and decreases the false positive rate. There is also another dimension, which is the In-Vehicle context, where the memory and computation power is limited, so the practical feasibility of any given CAN IDS needs to be evaluated in front of the constraints of the in-vehicle context.

3 In-Vehicle Intrusion Detection System

We propose a new IDS for vehicle described through 3 levels: 1) the CAN data level resumed in a feature matrix, 2) the time sequence modelling level and 3) the anomaly detection level. Notice that the training and update of the models are performed offline and outside from the vehicle; Only the exploitation of the models is embedded in the vehicle, which performs inference and detection.

3.1 Feature Matrix Construction

The first step is to transcript the flow of CAN messages, sent separately and asynchronously at irregular moments, into a Multirate Multivariate times series (MMTS) which contains both payload and flow information. The CAN data is composed of different messages broadcasted from different ECUs. The stream S of messages is represented by a sequence of N messages $S = \{V_1, ..., V_N\}$. We denote $V_i = \{id_i, P_i, t_i\}$ the i-th message captured by the IDS. $id_i \in \{1, ..., m\}$ is the ID of the ECU that sent the message, m is the number of ECU, P_i is the payload contained in the message and $t_i \in [0, t_N]$ is the time of reception. The payload P_i contains a vector of size n_{id} containing the signal sent from ECU id. The number of variables n_{id} extracted from each payload depends on the ECU.

Although deep learning models can be trained with this type of sequences, this representation is clearly not optimal for the learning of neural networks. The two main problems of this representation is that the payload variables from the same ECU are not split around the sequence and the time interval between two messages is not constant.

We change the messages flow representation S into a multivariate time series representation T. The time range of the messages flow is discretized into K time stamps of constant time Δ . The time series $T = \{t_1, ..., t_K\}$ is a series of time points, each time point represents the set of messages received during the corresponding time stamp $t_k = \{V_t | t \in [(k-1)\Delta, k\Delta[\}$. This time series is then represented by a matrix $M_{\sum_{id=1}^{m}(n_{id}+1)\times K}$ where each column represents a time point and the rows represent the variables contained in the payloads of all ECU. For each time point we could have received several messages from the same ECU and have different values for the same variables; In these cases we keep the last received values since we want to take our decision on the most recent available information. We also add to this matrix a row for each ECU indicating the number of messages received from this ECU. This representation regroups both payload and flow features, which enables the model to detect both attacks on payload and flow in the in-vehicle communication system.

3.2 CAN Sequence Modelling

Many DNNs architectures are used for time series modelling tasks. In this work, we review 4 types of sequence modelling architectures: FCN, CNN, LSTM and TCN. We compare their ability to learn the hierarchical representation vector of the CAN matrix to perform anomaly detection. This vector is either given by the bottleneck of the autoencoders in the unsupervised task, or the final layer of the architectures in the supervised task.

Fully-Connected Network (FCN): FCN is a standard architecture for deep learning models [2]. FCN is a generic architecture usable for any type of data. All the neurons in layer l_i receive the signal from every neuron in the layer l_{i-1} and send their output to every neurons of the layer l_{i+1} with $i \in [1, L]$ (L number of layers in the architecture). The elements of the time series are treated independently from each other, thus the temporal dimension of the data is ignored with this architecture.

Recurrent Neural Network (LSTM): Recurrent neural networks are explicitly devoted to sequence modelling [9]. They avoid the long-term dependency vanishing problem using cells state that is used as an internal memory. At each time, the network learn which information to add, to forget and to update into the cells state. Based on the cells state, inputs, previous hidden state, the LSTM learn a vector representation (hidden state) of the time series a the current time.

1D Convolutional Neural Network (CNN): In the context of time series, convolution is a sliding filter over the time series. The time series exhibits only one dimension. Thus this convolution will result in a moving average with a sliding window. Therefore, applying several filters results in learning several discriminative features which are useful for sequence modelling. Besides, the same convolution is used to find the result for all timestamps $t \in [1, T]$ (weight sharing). This is a valuable property, as it enables the model to learn filters that are invariant across the time dimension. 1D CNN for time series is characterized with a causal convolution; It means that the output at time t is convolved only with elements from time t or earlier in the previous layers. This characteristics ensures that the sequence input must have a one-to-one causal relationship in chronological order. The result of convolution can be considered as a time series whose dimension is equal to the number of these filters used. 1D CNN has another layer with pooling operation, which achieves dimension reduction of feature maps while preserving most information.

Temporal Convolution Network (TCN): TCN with dilated convolution is designed to combine simplicity and very long term memory [3]. There are many differences with 1D CNN described above. In addition to the causal convolution, the architecture can take a sequence of any length and map it to an output sequence with the same length. To achieve this, zero padding of length (kernel size - 1) is added. Moreover, the TCN architecture can look very far into the past using a combination of residual layers and dilated convolution. The dilated convolution [15] enables an exponentially sizeable receptive field using dilation factor d and the filter size k. When using dilated convolutions, we increase d exponentially with the depth of the network, allowing a very large history using deep networks.

3.3 Anomaly Detection

Unsupervised IDS. In this work, the autoencoder aims to reconstruct the input sequence (a multivariate time series). Formally, given a sequence $T = (t_1, ..., t_K)$ where $T_i \in \mathbb{R}^n$ and n is the number of variables, the autoencoder aims at predicting $\hat{T} = (\hat{t}_1, ..., \hat{t}_K)$ at each time (sequence-to-sequence problem). The autoencoder that performs a nonlinear mapping from the current state to its identity, is decomposed into two parts: the encoder and the decoder. The encoder projects the temporal pattern dependencies and trends of the time series in latent space h. The latent vector is given by $h^i = f(T.W^i + b^i)$, where W^i and b^i respectively denote the weight matrix and bias up to the bottleneck i-th layer. The decoder, considered as the transposed network of the encoder, uses the information of latent space h to reconstruct the input sequence: $\hat{T} = f(h^i.W_d^i + b_d)$. The mean square loss error (MSE) is used to perform an end-to-end learning objective: $L(T, \hat{T}) = \frac{1}{K}\Sigma_i^n(t_i - \hat{t}_i)^2$. At the inference phase, the MSE is used as an anomaly score. The idea is that the autoencoder learnt to reconstruct only the normal data and will obtain a high MSE on the anomaly.

Supervised IDS. Supervised IDS use a FCN to make anomaly prediction from the vector representation of the time series learnt from sequence modeling level. In this case, we suppose that the training dataset contains labelled attack examples and these attacks form an homogeneous class. These requirements generally hold when we construct a model that aims to detect well-known types of attack. Formally, we assume that there are 2 classes: Normal (0) and Anomaly (1). The learning set is a collection of pairs $\{(T_1, Y_1), ..., (T_K, Y_K)\}$ where T_i is a multivariate sequence and $Y_i \in \{0, 1\}$ is the corresponding label. The classifier training is performed by minimization of the cross entropy between the true class and predicted class. Notice that the classes are highly unbalanced, the anomaly is much smaller than the normal class, the classes are therefore weighted in the cross entropy in function on their prior. At the inference phase, the MLP returns the probability of anomaly.

4 Experiments and Results

4.1 SynCAN Dataset

SynCAN is a synthetic dataset proposed in [8]. The data consists of 10 different message IDs. We evaluate our model on the following types of attack: *Plateau* (a signal is overwritten to a constant value), *Continuous* (a signal is overwritten so that it slowly drifts away), *Playback* (a signal value is overwritten over a period of time with a recorded time series), *Suppress* (the attacker prevents an ECU from sending messages), and *Flooding* (the attacker sends messages of a particular ID with high frequency to the CAN bus).

We set five seconds as an estimated time-frame for the intrusion detection system to monitor the vehicle. Thus, the sampling window is fixed to $\Delta = 50ms$, and each sequence is composed of $K = 100$ consecutive elements. From a general standpoint, K and Δ are hyperparameters which depend on the domain expert requirement (the maximum memory size, forensic analysis and safety protocol to enable the prevention actions). The feature matrix size is (100*30) where 30 is the sum of 20 signal features and 10 occurrence features. We scale the data between $[0, 1]$ using min-max normalization.

A sequence is labelled normal if all elements in the sequence are normal. If a sequence contains at least one anomaly, the sequence is considered as an anomaly. SynCAN database is a collection of $\sim 2'000'000$ normal sequences. 70% of them are used for training and 10% for validation. The last 20% are mixed with anomalous sequences to build the test data. We have 5 test databases, one per attack, made of 70% normal examples and 30% anomalous examples.

4.2 Results

We use 4 different architectures (FCN, CNN, TCN, LSTM) with 2 experiment settings: unsupervised anomaly detection and supervised anomaly detection. We

have trained the models on 500 epochs, with a batch size 100. We used ADAM as the optimizer for the gradient descent with learning rate decay.[1]

In Table 1, we show the metrics on the unsupervised task using autoencoders with different architectures. All architectures show excellent performances for all types of attack. TCN is slightly better on most attack cases and comparable with LSTM on *Plateau* attack. Notice that on the *Suppress* attack, the models perform worse than on the other attacks, while the CNN collapses with a lot of false positive. It shows that *Suppress* attack is unobtrusive. Moreover, in representation matrix M, there is no explicit mention to the missing values. Nevertheless, TCN and LSTM still have good results, thus can implicitely retrieve the information in the learning stage from the representation matrix.

Table 1. Autoencoder-based architectures results

	TCN			LSTM			CNN			FCN		
	Precis.	Recall	F1	Precis.	Recall	F1	Precis.	Recall	F1	Precis.	Recall	F1
Continues	**0.997**	**0.991**	**0.994**	0.991	0.988	0.990	0.996	0.988	0.992	0.993	0.978	0.985
Plateau	0.995	0.984	0.990	**0.996**	**0.985**	**0.991**	0.993	0.979	0.986	0.990	0.981	0.987
Suppress	**0.986**	**0.957**	**0.971**	0.984	0.954	0.969	0.951	0.554	0.700	0.951	0.862	0.904
Flooding	0.995	0.986	0.991	0.996	0.988	0.991	**0.996**	**0.989**	**0.992**	0.996	0.988	0.991
Playback	**0.996**	**0.989**	**0.992**	0.996	0.986	0.991	0.994	0.989	0.991	0.995	0.988	0.991

In Table 2, we show the experiments on the supervised task (Binary classifier). TCN is still slightly better than the others, and close to the CNN. Notice that for LSTM architecture, the results are not as good as in unsupervised setting. Indeed, we have reduced the rate of normal data in the training set in order to rebalance the data and help the model not to learn from the normal features only. LSTMs are more data-hungry than CNN and TCN. It shows that TCN needs less data than LSTM for CAN data modelling.

In Table 3, we conduct the same experiment but we eliminate the occurrence features from representation matrix M. We notice for the *Flooding* attack, the performances of all the models decreases dramatically. Indeed this attack impacts the flow of the CAN data, and this information is encoded through the occurrences in the matrix. We also observe that the performances are slightly worse on the *Playback* and *Plateau* attacks. Therefore, payload-based attacks are also easier to detect when the occurrence features are present in the matrix. Hence full matrix, with both signal features and occurrence features, contributes to detect both payload and flow-based attacks.

Finally, in Table 4, we have compared the models in terms of training time and size of parameters. The latter reflects the memory needed by the IDS to work. Remind that the IDS are embedded in vehicle where memory resources is strongly limited. TCN is good both in terms of training time and model size. TCN benefits from filters shared weight, so it dramatically reduces the number of parameters. When the size of the input data is increasing, the number of

[1] See https://github.com/anonymeEG/Deep-Learning-4-IDS for implementation.

Table 2. Classification using the occurrence matrix representation

	TCN			CNN			LSTM			FCN		
	Precis.	Recall	F1	Precis.	Recall	F1	Precis.	Recall	F1	Precis.	Recall	F1
Continues	**0.996**	**0.986**	0.990	0.995	0.991	0.995	0.959	0.933	0.948	0.89	1.00	0.94
Plateau	**0.997**	**0.998**	0.997	0.991	0.953	0.971	0.984	0.953	0.968	0.87	1.00	0.93
Suppress	**0.999**	**0.999**	**0.999**	0.998	0.998	0.999	0.992	0.999	0.995	0.86	1.00	0.92
Flooding	**0.999**	**0.999**	**0.999**	0.999	0.999	0.999	0.999	0.999	0.999	0.87	1.00	0.93
Playback	**0.997**	0.988	0.992	0.997	0.995	**0.995**	0.994	0.989	0.991	0.89	1.00	0.94

Table 3. Classification using the standard sampling without occurrence features

	TCN			CNN			LSTM		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Continues	0.995	0.994	**0.994**	0.983	0.990	0.986	0.945	0.957	0.950
Plateau	0.973	0.978	0.975	0.995	0.998	**0.996**	0.959	0.984	0.971
Suppress	0.986	0.996	0.990	0.990	0.971	0.980	0.939	0.969	0.953
Flooding	0.985	0.978	0.981	0.971	0.915	0.928	0.927	0.972	0.913
Playback	0.992	**0.998**	0.994	0.992	0.772	0.868	0.935	0.924	0.929

Table 4. Models characteristics vs. computational resources

Models	Training time	Number of parameters	Model size (32-bits floats)
FCN	8022 s	75238	0,3 MB
CNN	10011 s	9518	0.03 MB
TCN	7969 s	3822	0.01 MB
LSTM	92714 s	2920	0.01 MB

parameters does not explode exponentially. Unlike LSTM, TCN convolutions can be done in parallel since the same filter is used in the layer. Therefore, in both training and inference phase, even though the series is long, it can be processed as a whole, whereas with LSTM, they must be processed sequentially.

5 Conclusion

In this paper, we introduce a novel in-vehicle intrusion detection system based on a large series of experiments which validate the different levels of the system: 1) At the data level, we use a representation matrix to structure the CAN data information that groups both flow and payload information. 2) At the sequence modelling level, we use a TCN architecture, since we have shown that it performs well with respect to the detection metrics and computational resources consumption. 3) At the detection level, we jointly use a classifier and an autoencoder, so the IDS can deal with both known and unknown attacks. Notice that our results were established by using the SynCAN Dataset, which is the only available public dataset as far as we know.

The in-vehicle system has many components, and we have implicitly assumed that the monitoring of the data was centralized. Nonetheless, in new secured in-

vehicle architectures, the parts of the system are isolated, so the CAN data topology changes, and we need to think about a distributed framework for IDS. Another important issue concerns the compression of deep learning models to better fit the embedded capacity of the in-vehicle system.

References

1. Avatefipour, O., Malik, H.: State-of-the-art survey on in-vehicle network communication (can-bus) security and vulnerabilities. CoRR (2018)
2. Bagnall, A.J., Bostrom, A., Large, J., Lines, J.: The great time series classification bake off: an experimental evaluation of recently proposed algorithms. extended version. CoRR abs/1602.01711 (2016)
3. Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. CoRR abs/1803.01271 (2018)
4. Dupont, G., den Hartog, J., Etalle, S., Lekidis, A.: Network intrusion detection systems for in-vehicle network - technical report. CoRR (2019)
5. Seo, E., Song, H.M., Kim, H.K.: Gids: gan based intrusion detection system for in-vehicle network. In: 2018 16th (PST) (2018)
6. Farsi, M., Ratcliff, K., Barbosa, M.: An overview of controller area network. Comput. Control Eng. J. **10**(3), 113–120 (1999)
7. Martinelli, F., Mercaldo, F., Nardone, V., Santone, A.: Car hacking identification through fuzzy logic algorithms. In: 2017 (FUZZ-IEEE), pp. 1–7 (2017)
8. Hanselmann, M., Strauss, T., Dormann, K., Ulmer, H.: An unsupervised intrusion detection system for high dimensional CAN bus data. CoRR (2019)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
10. Hoppe, T., Kiltz, S., Dittmann, J.: Security threats to automotive CAN networks - practical examples and selected short-term countermeasures. Reliab. Eng. Syst. Saf. **96**(1), 11–25 (2011)
11. Kang, M.J., Kang, J.W.: Intrusion detection system using deep neural network for in-vehicle network security. PLoS ONE **11**, 1–17 (2016)
12. Koscher, K., et al.: Experimental security analysis of a modern automobile. In: 31st IEEE S&P (2010)
13. Moore, M.R., Bridges, R.A., Combs, F.L., Starr, M.S., Prowell, S.J.: A data-driven approach to in-vehicle intrusion detection. In: CISRC '2017 (2017)
14. Nilsson, D.K., Larson, U., Picasso, F., Jonsson, E.: A first simulation of attacks in the automotive network communications protocol flexray. In: CI- SIS'2008 (2008)
15. Oord, V., et al.: A generative model for raw audio (2016)
16. Pawelec, K., Bridges, R.A., Combs, F.L.: Towards a CAN IDS based on a neural-network data field predictor. CoRR (2018)
17. Song, H.M., Woo, J., Kim, H.K.: In-vehicle network intrusion detection using deep convolutional neural network. Veh. Commun. **21**, 100198 (2020)
18. Young, C., Olufowobi, H., Bloom, G., Zambreno, J.: Automotive intrusion detection based on constant CAN message frequencies across vehicle driving modes. In: AutoSec (2019)
19. Song, H., Woo, J., Kang, H.: Automotive intrusion In-vehicle network, Controller area network (CAN), Intrusion detection, Convolutional neural network. VC (2020)