Towards a CAN IDS Based on a Neural Network Data Field Predictor*

Krzysztof Pawelec
The Pennsylvania State University
paweleckrzysztof1@gmail.com

Robert A. Bridges
Oak Ridge National Laboratory
bridgesra@ornl.gov

Frank L. Combs
Oak Ridge National Laboratory
combsfl@ornl.gov

ABSTRACT

Modern vehicles contain a few controller area networks (CANs), which allow scores of on-board electronic control units (ECUs) to communicate messages critical to vehicle functions and driver safety. CAN provides a lightweight and reliable broadcast protocol but is bereft of security features. As evidenced by many recent research works, CAN exploits are possible both remotely and with direct access, fueling a growing CAN intrusion detection system (IDS) body of research. A challenge for pioneering vehicle-agnostic IDSs is that passenger vehicles' CAN message encodings are proprietary, defined and held secret by original equipment manufacturers (OEMs). Targeting detection of next-generation attacks, in which messages are sent from the expected ECU at the expected time but with malicious content, researchers are now seeking to leverage "CAN data models", which predict future CAN messages and use prediction error to identify anomalous, hopefully malicious CAN messages. Yet, current works model CAN signals post-translation, i.e., after applying OEM-donated or reverse-engineered translations from raw data. We present initial IDS results testing deep neural networks used to predict CAN data at the bit level, targeting IDS capabilities that avoiding reverse engineering proprietary encodings. Our results suggest the method is promising for data with signals exhibiting dependence on previous or concurrent inputs.

CCS CONCEPTS

• **Security and privacy → Artificial immune systems**;

KEYWORDS

controller area network; CAN bus; in-vehicle security; anomaly detection; intrusion detection; neural network; deep learning

ACM Reference Format:
Krzysztof Pawelec, Robert A. Bridges, and Frank L. Combs. 2019. Towards a CAN IDS Based on a Neural Network Data Field Predictor. In *ACM Workshop on Automotive Cybersecurity (AutoSec '19), March 27, 2019, Richardson, TX, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3309171.3309180

AutoSec '19, March 27, 2019, Richardson, TX, USA

ACM ISBN 978-1-4503-6180-4/19/03…$15.00
https://doi.org/10.1145/3309171.3309180

1 INTRODUCTION & BACKGROUND

Modern vehicles are increasingly "drive-by-wire" meaning once-mechanical interfaces of subsystems have been replaced by communication of electronic control units (ECUs), or small computers orchestrating the subsystems. Rather than using dedicated connections for each ECU pair, a few controller area networks (CANs) allow broadcast communications of all ECUs. In particular, we focus on the high-speed (250Kbs-500Kbs) controller area network (CAN) bus, as it is used for much of critical vehicle communications.

Figure 1: CAN 2.0 data frame depicted. Image from Cho & Shin [4] of. There are two important fields, the Arbitration ID (AID) used for indexing and prioritizing frames and the data field containing up to 64 bits of message contents.

CAN 2.0 provides a protocol defining the physical and data link layers [1] (Figure 1). Each packet's information is contained in two fields, the Arbitration ID (AID) used for indexing and prioritizing frames and the data field containing up to 64 bits of message contents. The mapping of the data field's bits to the signals it encodes is a proprietary secret, defined by the original equipment manufacturers (OEMs, e.g., Ford, GM), and the encodings change depending on make, model, year, and even vehicle specifications. This poses an obstacle for producing *vehicle-agnostic* solutions for automotive CANs, in particular, defensive and offensive cyber security, which are desirable for many applications, in particular to existing vehicles. See recent work of Verma et al. [24], and Nolan et al. [19] on discovering the syntax and semantics of automotive CAN data.

CAN is a reliable and lightweight protocol, but it has few security features, e.g., no encryption nor authentication, and has been proven to be exploitable with direct access [2, 9, 15, 17] or even remotely [16, 25]. The attack surface for in-vehicle CANs is growing as cars become increasingly exposed e.g. via USB, cellular, bluetooth and the advent of vehicle-to-vehicle and -infrastructure networking. Providing effective intrusion detection for automotive CANs is a burgeoning research topic [14].

1.1 Related CAN IDS Works

Initial automotive CAN IDS research has been rule-based [9, 18], which pushes security to OEMs, as rules are dependent on CAN encodings (model-specific) and may require knowledge of specific attacks. Multiple works [7, 17, 21] exploit message frequency anomalies for vehicle-agnostic detection of message injection attacks. In response to the infamous Miller and Valesek remote Jeep hack [16] (which used a masquerade attack in which one ECU sent malicious

braking signals while the brake ECU was silenced), multiple efforts have proposed data-driven efforts for ECU identification to detect AIDs originating from the wrong transmitter [4, 5, 11].

The logical next-generation attack involves a reprogrammed ECU sending appropriate AIDs with appropriate timing, but with augmented, potentially malicious, data field contents. After-market "chipping" kits exhibit this capability by reprogramming ECUs, although in practice these are used for performance-tuning, not malicious purposes.

Works are emerging that test supervised deep learners trained on specific attacks with labeled data [10, 13]. We seek anomaly detection to avoid training towards a specific attack.

CAN IDS research into unsupervised learning methodologies for detecting malicious messages has begun modeling correlations inherent to the CAN data that are broken by attacks. Tyree et al. [23] propose a manifold learning technique to identify relationships in CAN data that are broken during attacks that do not coordinate related signals. Their technique requires at least the ability to tokenize (partition) the up-to 64-bit CAN data fields into signal-sized messages but not fully translate the CAN data. The other three works seeking to exploit correlations of signals in the CAN data and, thus, require complete knowledge of the modeled signals' encodings: Ganesan et al. [6] learn correlation of value pairs (e.g., speed, accelerator pedal position) using both CAN and sensor data to detect injection attacks. IDS research of Li [12] and of Testud [22] propose a three-step process to model CAN packets and detect unexpected packets: (1) reverse engineer or partner with an OEM to obtain many signals in the CAN data, (2) train deep learning, neural network regressor(s) to predict the next signal value(s) from the history of observations, (3) use the error in predicted values from observed as an online anomaly detector.

We present initial results for a CAN prediction model without step (1). That is, previous work translated the 64-bit data field into the signals it encodes (requiring OEM knowledge or tedious reverse engineering) and built models of the signals. Rather, our approach models an AID's 64-bit data field. Hence, we commence prediction and detection (steps (2) and (3)) without requiring any translation of the CAN message bits to signals.

1.2 Contributions

Our long-term goal is to provide an after-market IDS for ideally all passenger vehicles. This means we cannot rely on OEM-defined CAN mappings. En route to this goal we adopt the neural network CAN prediction model; specifically, from a history of CAN data our regressors predict the next CAN data field, and we too use prediction error to detect anomalous messages. Unlike the previous two similar works [12, 22], we do not translate CAN data fields to signals, as we do not have the OEM's proprietary mappings. Instead, we train a deep neural network for each AID to predict its next 64-bit data field. The primary contribution of this work is presentation of initial results showing efficacy of the bit-level CAN models for attack detection. The benefit of this approach is straight-forward—it extends the general CAN modeling frameworks for anomaly detection (which relies critically on OEM-proprietary CAN mappings) to a vehicle-agnostic detector, as no CAN mappings are assumed. Although our focus is CAN IDS, CAN models can be used for other applications, e.g. CAN simulators.

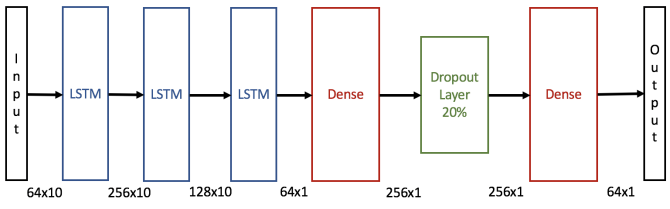

Figure 2: Neural Network architecture diagram depicted. Dimensions of the vector passed between layers given. Batch size was set to 32. Dropout between dense layers set to randomly ignore 20% of neurons in the first Dense layer.

2 CAN PREDICTION MODEL

The essential hypothesis of CAN prediction models is that there exists a dependency of future messages on recently passed or other concurrent messages. Hence, such an IDS will be blind to any attack manipulating data who's values are impervious to these dependencies or simply do not manipulating values, e.g., DOS by spamming, bus-off attacks [3].

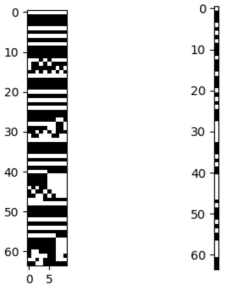

Figure 3: Training data example, on the left, ten consecutive signals are labeled by 11^{th} (right).

While our overall IDS is unsupervised—i.e., we do not require labeled attack and non-attack data—we exploit supervised learning to build a CAN prediction model. Specifically, we create labeled data by taking a fixed AID's most recently observed ten data fields and try to predict next (11^{th}). Hence, we model each AID independently.

Let $\mathbb{X} = \{x_i\}_{i=1}^{N}$ be the set of training examples and $\mathbb{Y} = \{y_i\}_{i=1}^{N}$ be the set of labels. Our training data is a tuple (x_i, y_i) where $x_i \in \{0,1\}^{10\times 64}$ and $y_i \in \{0,1\}^{64}$ as shown in Figure 3.

Recurrent neural networks (RNNs) model temporal/sequential dependence by including the previous prediction's hidden state as well as given inputs into the current prediction [20]. Long Short-Term Memory (LSTM) layers provide a particular architecture for portions of an RNN that seek to leverage dependence in modeling better than "vanilla" RNNs, as they are crafted to avoid vanishing gradient problems common in RNN training [8]. Hence, this statistical machinery is a natural choice for our model.

We build the model using Keras (www.keras.io), a Python deep learning module. See Figure 2. The model consists of three LSTM layers, a dropout layer, and two dense layers. The last layer having 64 nodes (one per predicted bit of the next data field) as the output and softmax as an activation function. Between the two dense layers, we include a dropout layer to prevent overfitting of our model. We set the layer's drop rate to 0.2; i.e., 20% of neurons in the first dense layer are dropped during training). To train the model

we used batch size of 32. Out of several tested architectures, where we varied the number/sizes of layers, this one was most accurate.

For each desired AID, we use the described LSTM on ambient CAN data collected during normal driving conditions. We denote such a model $\mathcal{M} = \mathcal{M}_{(\mathbb{X},\mathbb{Y},\text{AID})}$, where $\mathbb{X}$ is the set of training examples and $\mathbb{Y}$ is the set of labels for each example. For a given input vector $x_i \in \mathbb{X}$ (previous ten observed data fields), let $\mathcal{M}(x_i) = \hat{y}_i$ denote the predicted next 64-bit data field, $y_i \in \mathbb{Y}$.

To build an anomaly score from the AID's trained prediction model, we consider the error of each prediction, $e_i := \|y_i - \hat{y}_i\|_2$. To create the anomaly detector, after training, apply the model on the training data and compute the mean and variance of the observed prediction errors. Specifically, $\mu = \sum_{\mathbb{X}} e_i/N$, and $\sigma^2 = \sum_{\mathbb{X}}(e_i - \mu)^2/(N-1)$. Finally, we compute the Gaussian z-score of newly observed error $z_i = (e_i - \mu)/\sigma$ and use the one-sided p-value for our anomaly score, p-value(z) = 1 − CDF(z), where CDF is the Gaussian normal cumulative distribution function. Note that if the error is less than expected ($z < 0$) p-value$(z) > 0.5$ and p-value$(z) \to 1$ as $z \to -\infty$. Similarly, if the error is greater than expected ($z > 0$) p-value$(z) < .5$, and p-value$(z) \to 0$ as $z \to \infty$. Hence, a small p-values occurs if and only if the error is large relative to observations in training.

3 EXPERIMENT

We present two indicative experiments. The first is a model of an AID that (we believe) communicates the four wheels' speeds—four signals that vary little between each other and individually all move "continuously", meaning limited variance from one message to the next. The second AID includes a binary reverse light indicator, which need not depend on its past values and we cannot say whether it depends on other bits in the AID. Our understanding is from manual reverse engineering.

For data collection we used the Vehicle Spy software, produced by Intrepid Control Systems, Inc. (www.intrepidcs.com/products/software/vehicle-spy) allowing passive monitoring of CAN data via the OBD-II port. For training, we used a portion of CAN data recorded during ambient driving lasting 141 seconds. See Figure 4.

To test the detector, we inject CAN frames with each AID, separately, to emulate attacks on the CAN. It is important to stress that the anomaly detector does not consider the frequency nor the timestamp of CAN frame, only the sequence of data fields; hence, the high frequency injections emulate an ECU that is sending messages with false content. For each emulated attack (one per AID), we used an Arduino board for injecting CAN frames as well as the Vehicle Spy for recording CAN data, both connected to the vehicle via an OBD-II port. Reverse engineering of the signals allowed for physical verification that the attacks were effective. Note that our reverse engineering of these signals is not necessary for the model, but for understanding efficacy of the experiment/model.

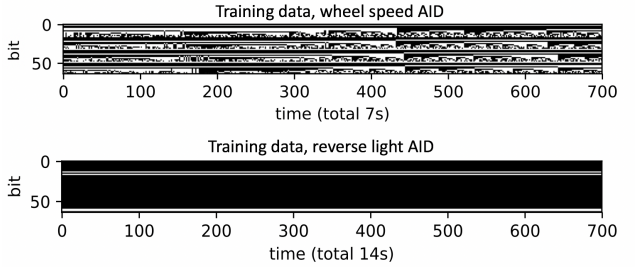

Figure 4: Time snippet of training data for two AIDs.

3.1 Wheel Speed AID

The actual attack happened from 14s to 29s of the trip. During that time the "attacker" repeatedly injected the same AID with the same message in the 64-bit data field. As can be seen in Figure 5, the p-value of the observed signals occurring between 14s to 29s is extremely low.

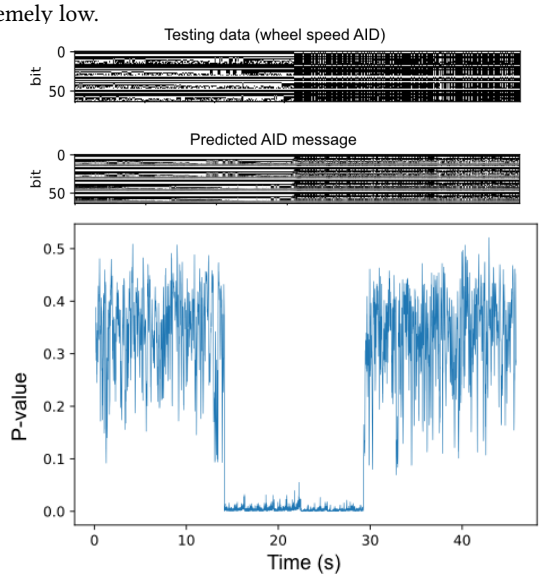

Figure 5: (Top) Time snippet of wheel speed AID testing data non-attack period (the left half) and the attack period (the right half). (Middle) Plot depicts the model's prediction. (Bottom) P-value anomaly score depicted for wheel speed AID. Attack period: 14-29s.

3.2 Reverse Lights AID

The actual attack happened from 14.5s until 29s of the capture. During that time the "attacker" repeatedly injected the same AID with the same message in the 64-bit data field. Referring to Figure 6, it is important to note that the p-value of the observed signals is extremely low throughout the test set. However, it hits actual 0 during the attack period.

3.3 Results Discussion

Overall, we have a very strong difference in our anomaly score between attack and non-attack periods, but finding an a priori threshold seems problematic. We conjecture that the current architecture is a better model for "continuous" signals with many distinct 64-bit messages (as in Figure 5), that move in a a clear pattern (e.g., as speed increases, the 2^0 place bit increases from 0 to 1, then the 2^1 place bit increases from 0 to 1, ...). Presumably the model is also learning co-variation between the four wheel speeds, which are similar values except when under attack. The second AID communicating seemingly binary signals is, unsurprisingly, harder

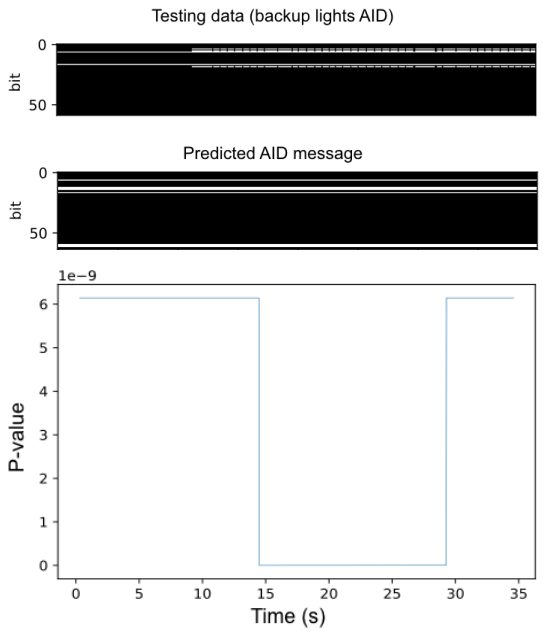

Figure 6: (Top) Time snippet of wheel speed AID testing data. Non-attack period occurs for roughly the first quarter and the attack period in the other three quarters. Top plot depicts actual data. (Middle)Plot depicts the model's prediction. (Bottom) P-value anomaly score depicted for reverse light AID. Attack period: 14.5-29s.

for the model to predict. Perhaps taking inputs from a variety of other AIDs may enhance prediction accuracy.

4 CONCLUSIONS & FUTURE WORK

Recent approaches to build CAN IDSs train a "CAN language model", that is, a machine learning model that can accurately predict the next CAN message from previous or concurrent messages. Previous works have trained models on reverse engineered signals, requiring OEM-proprietary (secret) knowledge. In this paper we build a CAN model at the bit level, eliminating the need for CAN data translation and present initial results in use for an IDS. One advantage of this approach is application to existing vehicles, and could, with sufficient development, be deployed as an after-market OBD-II plug-in. We model the 64-bit data field for each AID and use prediction error to identify anomalies. Our results suggest that AIDs encoding data signals that move "continuously" or in conjunction with each other yield strong detection results for attacks that manipulate these signals. The results are less compelling for signals that exhibit less dependencies on their past and possibly on other portions of the data frame.

For future work, we would like to refine the architecture of the neural network to more accurately predict non-malicious messages. We note that preliminary testing with alternate neural network configurations yielded less accurate results, but lends credence to future work aimed at optimizing the architecture for CAN modeling. Additionally, construction of a model that handles more than just one AID at a time will presumably increase accuracy as CANs communicate states of many different but physically related subsystems. Finally, work is emerging to automatically discover encoded signals in the CAN data fields (e.g, [19, 24]). The next step is to train the CAN models conditioned on information from these works.

ACKNOWLEDGEMENTS

Authors thank S. Hollifield, J. Laska, and M. Verma for fruitful discussions. Research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U.S. Department of Energy and the National Science Foundation Math Science Graduate Internship.

REFERENCES

[1] Robert Bosch GmbH. 1991. CAN Specification Version 2.0. (1991).
[2] Stephen Checkoway and others. 2011. Comprehensive Experimental Analyses of Automotive Attack Surfaces.. In USENIX Security Symposium. San Francisco.
[3] Kyong-Tak Cho and Kang G Shin. 2016. Error handling of in-vehicle networks makes them vulnerable. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 1044–1055.
[4] Kyong-Tak Cho and Kang G Shin. 2016. Fingerprinting Electronic Control Units for Vehicle Intrusion Detection.. In USENIX Security Symp. 911–927.
[5] Wonsuk Choi and others. '18. Identifying ECUs through Inimitable Characteristics of Signals in Controller Area Networks. IEEE Trans. Vehic. Tech. ('18).
[6] Arun Ganesan, Jayanthi Rao, and Kang Shin. 2017. Exploiting Consistency Among Heterogeneous Sensors for Vehicle Anomaly Detection, In SAE Technical Paper. (03 2017). https://doi.org/10.4271/2017-01-1654
[7] Mabrouka Gmiden, Mohamed Hedi Gmiden, and Hafedh Trabelsi. 2016. An intrusion detection method for securing in-vehicle CAN bus. In Proc. of Sciences and Techniques of Automatic Control and Computer Engineering. IEEE.
[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. Neural Comput. 9, 8 (Nov. 1997), 1735–1780. DOI: http://dx.doi.org/10.1162/neco.1997.9.8.1735
[9] Tobias Hoppe and others. 2008. Security threats to automotive CAN networks–practical examples and selected short-term countermeasures. In Inter. Conf. Comp. Safety, Reliability & Security. Springer.
[10] Min-Joo Kang and Je-Won Kang. 2016. Intrusion detection system using deep neural network for in-vehicle network security. PloS one 11, 6 (2016), e0155781.
[11] Hyunsung Lee, Seong Hoon Jeong, and Huy Kang Kim. 2017. OTIDS: A Novel Intrusion Detection System for In-vehicle Network by using Remote Frame. In PST (Privacy, Security and Trust). (accepted).
[12] Jun Li. 2016. Deep Learning on CAN Bus. (2016). https://youtu.be/1QSo5sOfXtI
[13] George Loukas and others. 2018. Cloud-Based Cyber-Physical Intrusion Detection for Vehicles Using Deep Learning. IEEE Access 6 (2018). DOI: http://dx.doi.org/10.1109/ACCESS.2017.2782159
[14] George Loukas, Eirini Karapistoli, Emmanouil Panaousis, Panagiotis Sarigiannidis, Anatolij Bezemskij, and Tuan Vuong. 2019. A taxonomy and survey of cyber-physical intrusion detection approaches for vehicles. Ad Hoc Networks 84 (2019), 124–147.
[15] Charlie Miller and Chris Valasek. 2013. Adventures in automotive networks and control units. Def Con 21 (2013), 260–264.
[16] Charlie Miller and Chris Valasek. 2015. Remote exploitation of an unaltered passenger vehicle. Black Hat USA 2015 (2015), 91.
[17] Michael R Moore, Robert A Bridges, Frank L Combs, Michael S Starr, and Stacy J Prowell. 2017. Modeling inter-signal arrival times for accurate detection of CAN bus signal injection attacks: a data-driven approach to in-vehicle intrusion detection. In Proc. CISRC. ACM, 11.
[18] Michael Müter and others. 2010. A structured approach to anomaly detection for in-vehicle networks. In IAS. IEEE.
[19] Brent Nolan and others. 2018. Unsupervised Time Series Extraction from Controller Area Network Payloads. In Proc. IEEE CAVS. (to appear).
[20] David E. Rumelhart and others. 1986. Learning representations by back-propagating errors. Nature 323 (Oct. 1986). dx.doi.org/10.1038/323533a0
[21] Hyun Min Song, Ha Rang Kim, and Huy Kang Kim. 2016. Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. In ICOIN. IEEE.
[22] Jean-Christophe Testud. 2017. Detecting ICS Attacks Through Process Variable Analysis. (2017). https://youtu.be/b4lut5uWs2w
[23] Zachariah Tyree, Robert A Bridges, Frank L Combs, and Michael R Moore. 2018. Exploiting the Shape of CAN Data for In-Vehicle Intrusion Detection. In IEEE CAVS. (to appear), arXiv:1808.10840.
[24] Miki E Verma, Robert A Bridges, and Samuel C Hollifield. 2018. ACTT: Automotive CAN Tokenization and Translation. In Proc. IEEE CSCI. (to appear) arXiv:1811.07897.
[25] Samuel Woo and others. 2015. A practical wireless attack on the connected car and security protocol for in-vehicle CAN. Tran. Intel. Trans. Sys. 16, 2 (2015).