2016 15th IEEE International Conference on Machine Learning and Applications

Automating ECU Identification for Vehicle Security

Michael Jaynes
and Dr. Ram Dantu
Department of Computer Science and Engineering
University of North Texas
Denton, Texas 76203
Email: michaeljaynes@my.unt.edu
rdantu@unt.edu

Roland Varriale II
and Nathaniel Evans
Cyber Operations, Analysis and Research
Argonne National Laboratory
Lemont, IL
Email: rvarriale@anl.gov
nevans@anl.gov

***Abstract*—The field of vehicular cybersecurity has received considerable media and research attention in the past few years. Given the increasingly connected aspect of consumer automobiles, along with the inherent danger of these machines, there has been a call for experienced security researchers to contribute towards the vehicle security domain. The proprietary nature of Controller Area Network (CAN) bus messages, however, creates a barrier of entry for those unfamiliar, due to the need to identify what the messages on a given vehicle's bus are broadcasting. This work aims to automate the process of correlating CAN bus messages with specific Electronic Control Unit (ECU) functions in a new vehicle, by creating a machine learning classifier that has been trained on a dataset of multiple vehicles from different manufacturers. The results show that accurate classification is possible, and that some ECUs that broadcast similar vehicle dynamics broadcast similar CAN messages.**

***Index Terms*—automotive security, machine learning, classification, CAN bus**

I. Introduction

As consumer automobiles have become increasingly dependent upon technology, interest in researching these vehicles as a potential attack surface has risen sharply. This is especially true given that some of the newer model years have been shipped with outward-facing channels of communication like Wi-Fi and 3G installed. This communication functionality enables remote cyber attacks through vulnerabilities that exist on vehicles, which are no longer strictly theoretical in nature. Dr. Charlie Miller and Chris Valasek have famously demonstrated some exploits of these vulnerabilities [1] [2] [3]; moreover, other researchers have provided substantial contributions within this domain [4]. To prepare for, and defend against, this threat, security researchers at vehicle manufacturers, government agencies, and research institutions are exploring many facets within the automotive cybersecurity field.

Cognitive distance provides a substantial roadblock which prevents more experienced security experts from applying their knowledge and skillset to the automotive domain. Simply put, although there are basic networking principles at work within vehicles, the difficulty in understanding the protocols and identifiers on a new vehicle lie in the proprietary nature of each vehicle's configuration. Consumer automobiles sold in America starting with model year 2008 all use a CAN bus architecture for communicating between the various embedded systems, or ECUs, that aid or control vehicle functions [5]. Despite the fact that the CAN bus is the standard for American cars, the actual messages that are sent on the CAN bus vary from manufacturer to manufacturer and even between different models and model years from the same manufacturer. This diversity provides a frustrating experience, as it is often unclear what any of the messages on the CAN bus are broadcasting when the traffic is monitored — particularly if the vehicle is unfamiliar or the researcher is less experienced. The lack of consistent information creates a myriad of problems directly affecting ability to create sound, applicable, and secure solutions, since the ECU communication data is not publicly known.

Previous research exists that demonstrates the validity of utilizing machine learning classifiers on vehicle data, as is presented in this paper. A recent SAE technical paper [6] used a neural network-based recgonition module in order to recognize manipulated ECUs. The recognition module uses simulated vehicle data such as speed and RPM as input into a neural network and was able to recognize ECUs that had been manipulated. Similarly, an older paper [7] utilized other features about driver behaviors, such as trip distance and average and maximum acceleration and deceleration, which are fed to a machine learning algorithm for the purpose of saving fuel by optimizing power control. The authors also utilize a neural network specifically for the purpose of predicting road types and traffic congestion in order to further improve the reduction in fuel usage.

The current methods of determining the identifiers and message contents of CAN bus messages can be time-consuming, unreliable, or infeasible. Currently, identifying CAN messages involves a manual process of monitoring CAN traffic while performing vehicle operation to produce a loose guideline to send messages using trial-and-error. For example, a researcher might watch the CAN bus traffic and look for messages that change when turning the steering wheel, or pressing the brake pedal. Unfortunately, this will not always allow for a researcher to distinguish between closely-related messages, and there are a fair number of messages that cannot be identified in this manner. Another method of identification is to physically disconnect individual ECUs from a CAN bus

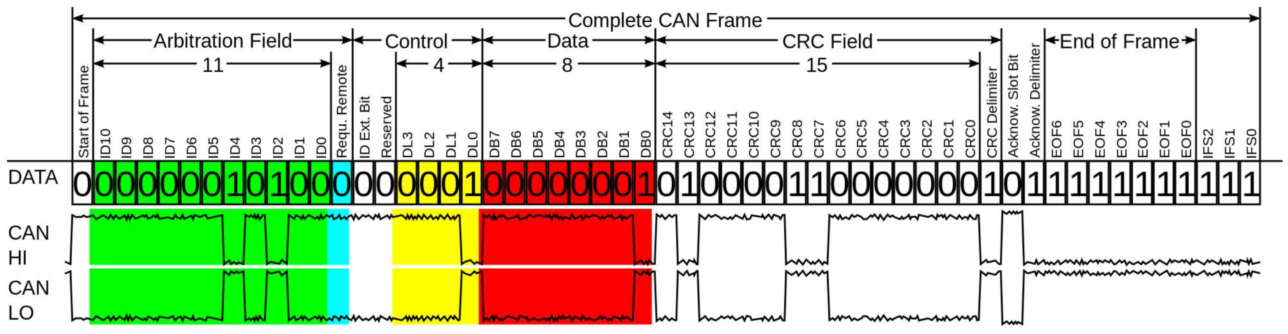

Fig. 1. CAN-frame format

and look for the absence of messages which were previously present. Since many ECUs regularly send out CAN messages with specific arbitration identifiers (arbIDs), disconnecting an ECU can determine exactly which messages that ECU is sending out. However, this method is much more difficult and might be infeasible for many researchers.

Instead of being forced to use tedious, manual methods, the goal of this research was to determine if machine learning techniques could aid in an automated identification of ECUs within the context of CAN bus message content. Due to the restrictions of the CAN protocol, each CAN message contains only 8 bytes of data. Although ECU identifiers may differ greatly on different vehicles, some messages seem to have similar message formats. Classifiers such as nearest neighbor (*k*-NN) and multilayer perceptron networks have previously provided accurate results on similar types of data sets, given a sufficient training set of labeled vehicle CAN data. Initially this data would have to be manually labeled using one of the above methods, and a variety of different vehicles and manufacturers should be included to ensure that the new vehicle in question would be covered. Eventually, a user front-end could be designed so that newly classified results could be saved, verified, and added to the dataset. This would continue to increase the accuracy of and eventually even reduce the need for the machine learning process entirely. For testing purposes, one vehicle of the dataset of collected vehicle data would be kept out of training the classifier to serve as the test set similar to the concept of 10-fold cross validation.

II. Dataset Collection

Data collection for each vehicle included in the dataset was obtained through Intrepid Control Systems' neoVI FIRE CAN bus hardware, which plugs into a vehicle's on-board diagnostic (OBDII) port and provides a stream of data to a laptop over USB. However, it would be possible to perform the same data collection with any number of much less expensive hardware solutions. Vehicle Spy 3, a software product also distributed by Intrepid[1], provided a graphical user interface that offers substantial functionality, such as scripting, logging, and filtering features that make it easier to sort through the myriad CAN bus messages, in addition to the ability to perform basic operations, such as reading and transmitting messages. Although this study explicitly used the proprietary offerings of Intrepid, there are also open-source software solutions, such as the SocketCAN can-utils on Linux, which may be coupled with cheaper hardware to offer similar functionality at a lower cost.

A complete set of messages for each vehicle were not identified, but a small subset was chosen in order to determine the viability of this method of identification. It was not feasible to identify a large number of messages in each vehicle, due to constraints on access to each vehicle. The messages that were chosen were brake pedal pressure, acceleration pedal pressure, steering wheel position, wheel speeds, and gear shift position. These were chosen mostly for the ease and consistency with which they could be identified by a researcher sitting in a new car for a short amount of time. After these messages had been identified and labeled in the Vehicle Spy 3 software, a log file is collected of a short (3-5 minutes) driving scenario that includes reversing, turning, accelerating, and braking. This log is then exported for pre-processing. Currently, the dataset includes log files from 9 different vehicles, ranging in model years from 2011-2016, including but not limited to vehicles from Ford, Toyota, Dodge, Hyundai, and VW. This selection of vehicles is representative of many of the vehicles present on US roadways. This diversity in the dataset suggests the content of the message is a more important factor in identification than manufacturer.

III. Feature Selection

The broadcast nature of the CAN bus limits the set of features available for message classification, as there are no source or destination addresses, nor is there much header information at all. The CAN format includes an arbID for each message which serves as something of an identifier for the message — however, different ECUs might both send messages with the same arbID, and every device on the bus receives every message broadcast. Further considerations of the CAN bus messaging protocol, such as the fact that certain 1-bit flags that are available as part of the CAN specification are rarely different between any of the messages on the CAN bus of an average car, can be omitted completely due to the

[1]https://www.intrepidcs.com/

lack of information gain for that feature. For reference, Fig. 1 shows the format of a CAN frame. The Control bits typically are the same for all messages on a particular vehicle, and the CRC data doesn't provide much information.

Initially this study utilized the following features:

- Arbitration ID (arbID)
- Relative time since last message received on that arbID (Δt, in seconds)
- Message data payload (separated into 8 bytes, B1-B8)

The Δt attribute was provided by the Vehicle Spy 3 software; no extra work was required to calculate this value. This value was included as a feature because it was hypothesized that certain types of messages might share a timing setup on different vehicles. However, once a sizeable dataset had been collected and different classifiers were being tested, classification was tested with the first two attributes removed (arbID and Δt). For most classifiers, this produced results that had similar or even higher accuracy values, which suggests that for some of the messages being identified, there is enough information gain in the data payload of the messages themselves for identification purposes.

IV. Classification

The training and testing of machine learning algorithms for the purposes of classifying this CAN bus data was done with the open-source, Java-based program Weka. The use of Weka allows for operation in diverse environments since it is written in Java and can utilize the trained classifier in a potential front-end by calling on Weka libraries across a diverse pool of devices and operating systems.

For pre-processing this data, a Python script was written and used alongside a standard text editor. The text editor was used to remove extraneous leading lines put into the log file by Vehicle Spy 3 and to replace missing byte values with a value of zero. The script then read through the logs, converted the hexadecimal (hex) representation of the bytes into decimal, and removed lines that were unlabeled messages. This was done to reduce non-operational data and increase overall processing and classification speed of a reduced dataset. The hex to decimal conversion was done to handle Weka's difficulty with parsing hex values the bytes needed to be numeric and not nominal attributes. Finally, the individual logs were combined into a larger training file and shuffled, with one vehicle's log held back to be used as the test set.

A number of different classifiers were applied and then the results were compared, trying to determine which would be best-suited to the dataset. Since the initial tests included the arbID as a feature of the model, classifiers based on decision trees seemed unlikely to produce accurate identification, since a decision tree would put a lot of weight on arbIDs while training, but the test vehicle will almost always be using different values for each message. The nearest neighbor classifier stood out as a potentially successful classifier, given the nature of CAN bus messages in most vehicles. Since there are only 8 bytes to work with, messages that perform similar functions in different vehicles typically use similar methods of scaling and

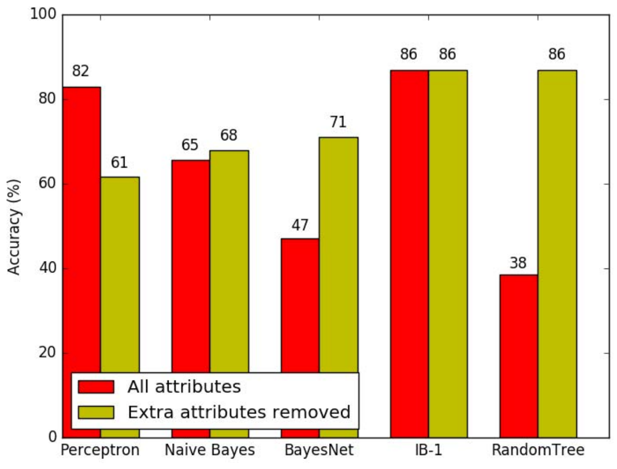

Fig. 2. Comparison of Weka classifiers

formatting. Therefore, even with very different arbID values, the data itself should be similar enough across vehicles that the correct class will be the closest in most distance metrics.

The following classifiers were tested, given by their Weka naming convention (note that some were omitted from section V):

- Naïve Bayes (with kernel estimation)
- Bayesian Network
- Simple Logistic
- Multilayer Perceptron
- LibSVM
- IB-k nearest neighbor
- KStar
- RandomForest decision tree

V. Results

Fig. 2 shows a breakdown of some of the accuracy values provided by different classifiers that were tested. This shows the similarity or dissimilarity between the accuracy before and after the arbID and Δt attributes were removed. In particular, the decision tree classifier performed very poorly when these values were left in, but performed quite well when they were removed. An increase in accuracy was expected with the removal of arbIDs, but the difference between the two was quite drastic. Meanwhile, the nearest neighbor classifier ignores the Δt and arbID values when determining distance anyway, so there is no change in accuracy when those values are removed.

Fig. 3 and Tables I and II provide more detailed information for the IB-1 nearest neighbor classifier, as that classifier had the most consistent performance, both with and without the additional attributes. Fig. 3 shows that certain classes were more easily classified by this model than others; in particular, the model exhibited poor performance identifying the steering wheel and accelerator messages. This can be seen in more detail in Table I. The steering wheel message had much fewer instances in the test set than some of the other messages, which

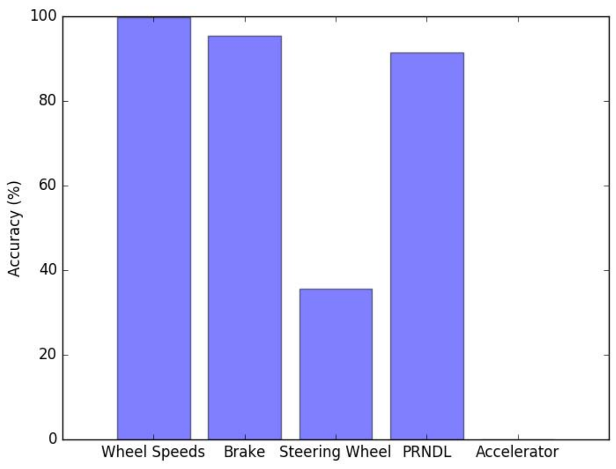

Fig. 3. Accuracy by message class, IB-1 classifier

TABLE I
IB-1 CONFUSION MATRIX

Classified as ->	Wheels	Brake	Steering	PRNDL	Accel
Wheels	13491	19	22	0	0
Brake	188	6777	79	30	20
Steering	138	72	116	0	0
PRNDL	0	3	0	149	11
Accel	0	873	1429	233	0

TABLE II
IB-1 PRECISION, RECALL, AND F-MEASURE

	TP Rate	FP Rate	Precision	Recall	F-Measure
Value	86.8%	3.7%	82.5%	86.8%	84.4%

might have contributed to the poor performance. However, the accelerator pedal had many more instances but the model still struggled to recognize these messages correctly, instead labeling them as steering wheel or brake messages.

This particular implementation of a nearest neighbor classifier is based on the work presented in [8]. It allows the user to specify an arbitrary *k* value or select from a few different nearest neighbor search algorithms (brute force search, KDTree [9], CoverTree [10]). The results presented here utilize the default settings of using the brute force search with a *k* value of 1.

VI. CONCLUSION

Given the high precision and recall for most of the classes, these results suggest the viability of the work, as well as areas of improvement as the dataset expands. Three out of the five classes tested had TP rates above 90%, and the overall FP rate was very low. One limitation of the dataset was that the accelerator pedal message was unable to be identified for every vehicle, and was therefore underrepresented in the training dataset. This seems to be reflected in the poor results for that specific message. Nevertheless, the results suggest that there is enough similarity between how certain messages (such as wheel speeds and brake pedal messages) are formatted on different vehicles, even across manufacturers, that as this dataset expands to include more vehicles and different types of messages, classification will still be possible.

VII. FUTURE WORK

In refining the approach to this problem, a larger set of messages for each vehicle will need to be identified. Expanding the diversity of message types that are included in the training process will further test the viability of this identification method, and provide insight into other weaknesses in this approach. Further testing will also involve using a trained model to classify the log file of a test vehicle with all of the messages present to determine if the model struggles with falsely identifying unidentified but similar messages on the CAN bus. Eventually, after the dataset includes multiple vehicles that have had a majority of their CAN messages identified, a front-end could be designed to allow a user to upload his or her own log file for classification. Once this front-end is developed, the process of identifying CAN messages on a new vehicle will be made much easier. Rather than having to spend hours in a vehicle analyzing how the messages on the CAN bus respond to different triggers or removing physical ECUs from the CAN bus, a user will be able to collect a short set of driving data and upload it for automated classification.

ACKNOWLEDGMENTS

The authors would like to thank the National Science Foundation (grant number 1229700) for partially funding this research.

REFERENCES

[1] C. Miller and C. Valasek, "Adventures in automotive networks and control units," *DEF CON*, vol. 21, pp. 260–264, 2013.
[2] ——, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, 2015.
[3] ——, "A survey of remote automotive attack surfaces," *black hat USA*, 2014.
[4] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno *et al.*, "Comprehensive experimental analyses of automotive attack surfaces." in *USENIX Security Symposium*. San Francisco, 2011.
[5] I. Standard, "Iso 11898, 1993," *Road vehicles–interchange of digital information–Controller Area Network (CAN) for high-speed communication*, 1993.
[6] A. Wasicek and A. Weimerskirch, "Recognizing manipulated electronic control units," SAE Technical Paper, Tech. Rep., 2015.
[7] J. Park, Z. Chen, L. Kiliaris, M. L. Kuang, M. A. Masrur, A. M. Phillips, and Y. L. Murphey, "Intelligent vehicle power control based on machine learning of optimal control parameters and prediction of road type and traffic congestion," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 9, pp. 4741–4756, 2009.
[8] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Machine learning*, vol. 6, no. 1, pp. 37–66, 1991.
[9] A. W. Moore, "An intoductory tutorial on kd-trees," 1991.
[10] A. Beygelzimer, S. Kakade, and J. Langford, "Cover trees for nearest neighbor," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 97–104.