Reverse Engineering Controller Area Network Messages using Unsupervised Machine Learning

Uchenna Ezeobi, Habeeb Olufowobi, Clinton Young, Joseph Zambreno, Gedare Bloom

***Abstract*—The smart city landscape is rife with opportunities for mobility and economic optimization, but also presents many security concerns spanning the range of components and systems in the smart ecosystem. One key enabler for this ecosystem is smart transportation and transit, which is foundationally built upon connected vehicles. Ensuring vehicular security, while necessary to guarantee passenger and pedestrian safety, is itself challenging due to the broad attack surfaces of modern automotive systems. A single car contains dozens to hundreds of small embedded computing devices known as electronic control units (ECUs) executing 100s of millions of lines of code; the inherent complexity of this tightly-integrated cyber-physical system (CPS) is one of the key problems that frustrate effective security. We describe an approach to help reduce the complexity of security analyses by leveraging unsupervised machine learning to learn clusters of messages passed between ECUs that correlate with changes in the CPS state of a vehicle as it moves throughout the world. Our approach can help to improve the security of vehicles in a smart city, and can leverage smart city infrastructure to further enrich and refine the quality of the machine learning output.**

I. Introduction

The notion of smart city includes smart transportation and the management of the infrastructure that supports its operations. Connected vehicles are important technologies supporting these services. The security issues around the use of connected vehicles in smart city ecosystems are becoming increasingly prominent due to the rising complexity of vehicles as a complex, integrated cyber-physical system (CPS) [1]. A CPS is described by the intercommunication between the computational devices and their physical environment. A modern vehicle is a CPS consisting of hundreds of electronic control units (ECUs) to improve the efficiency of the vehicles in terms of safety, automation, and comfort. These ECUs also stand between the vehicle controls and the outside world, including other connected vehicles and smart city infrastructure as depicted in Figure 1. The communication among these ECUs is facilitated by the in-vehicle network to achieve personalized vehicle configuration and enhanced connectivity. However, these connectivities expose the vehicular systems to several emergent vulnerabilities, cyber threats, and physical compromise. An essential design consideration of the smart city is to account for the security of the computational components of the CPS vehicles within their broader ecosystem as depicted in Figure 2.

The adoption of the controller area network (CAN) bus for communication between the ECUs has led to the complexity of network attacks on connected vehicles. Researchers have demonstrated different attack surfaces that can be used to compromise vehicle operations remotely in modern vehicles due in large part to the integration with

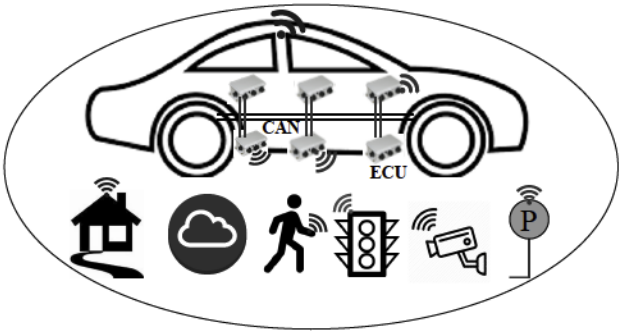

Fig. 1: In-vehicle security impacts the integrated smart city ecosystem.

CAN bus [2]. A plethora of approaches have been proposed for securing the CAN bus, which lacks security by design [3], [4]. The CAN bus has a standardized message transmission and operation, but the implementation varies for different manufacturers and vehicle models, make, or even trim. A de facto standard way to describe a specific implementation is with a CAN database file known as DBC (after its extension, .dbc). The DBC contains the translation for all CAN messages in a particular vehicle, and it is typically held secret by manufacturers and varies by implementation.

Algorithmic vulnerability analysis and attack detection, e.g., with supervised machine learning, requires the analysis of raw CAN messages by manually inspecting high volumes of data to reverse engineer message syntax and semantics to reconstruct the vehicle operations and contextualize the messages. Although network traffic analysis and packet recognition exists for enterprise networks and Internet protocols, the prior work in this area are inapplicable to the automotive network due to differences in the CAN protocol. Most important is that CAN messages do not include source nor destination addresses nor port numbers. Enterprise networks also have a clear separation between network and application layers which is not the case with CAN, which only uses the physical and link layers of the traditional stack.

Our goal, therefore, is to facilitate analyzing and reverse engineering the relationship between CAN messages and vehicular functions without extensive manual testing on a physical vehicle and without the use of a DBC file. To achieve this goal, we use an approach that focuses on reverse engineering and classifying messages transmitted in the CAN bus with minimal domain expertise. The reverse engineering process is required by researchers to appropriately map vehicular functionalities to their respective messages and understand the impact on the vehicular operation when altered. The typical approach to reverse engineer CAN data is to inject captured messages into the vehicle to see how it responds. This requires a physical vehicle in a controlled (lab) environment to be done safely. Our insight is those high-level descriptions of captured CAN data may provide coarse-grained labels on time intervals that relate them to vehicle operations. Key here is that some mapping must exist between time and operations, for example, at time x the vehicle turns left. In a smart city we envision this mapping could be facilitated through the vehicle-infrastructure communications, or even by infrastructure monitoring (cameras and other sensors) alone. The classification of messages is accomplished by leveraging unsupervised machine learning using clustering methods as the primary method to classify messages and their contents.

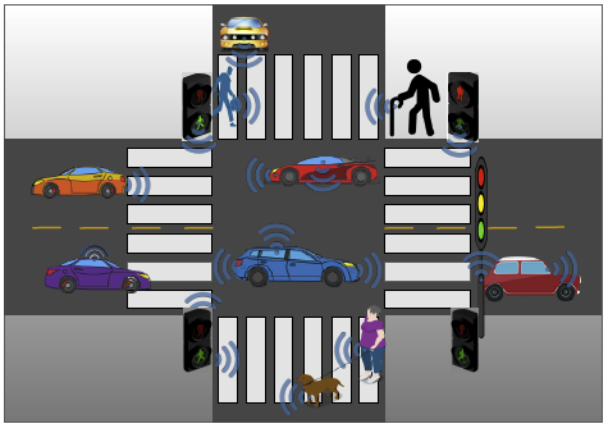

Fig. 2: CPS in a Smart City Ecosystem.

In a smart city ecosystem, the mapping of the message allows the correlation of the CPS state to the physical events—higher-level understanding of events to the lower-level functions—that are facilitated by the smart city infrastructure. The proposed reverse engineering approach can be used to characterize traffic by vehicle types, conjecture how events are generated, and facilitate the link between cyber communications and physical behavior of the externally visible changes in connected vehicles, such as the actions of a left turn, right turn, and break with their corresponding sequences of internal CAN messages.

The contributions of this paper are as follows:

- An approach to reverse engineer CAN using unsupervised learning
- Comparison of 4 clustering algorithms to correctly classify real CAN data.

II. Background And Related Work

The CAN bus is a serial communication protocol that consists of a set of nodes called ECUs. These ECUs are interconnected by a broadcast channel

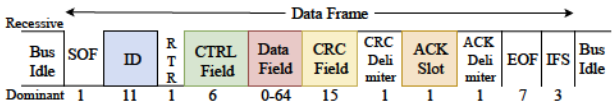

Fig. 3: Base CAN data frame format.

to transmit messages related to their functions. Vehicles utilize CAN as a serial communication protocol. The adoption of CAN reduces network complexity and wiring costs by simplifying mechanical connections or point-to-point wiring. Messages sent on the bus are broadcast to all the nodes communicating on the network. CAN implements the carrier sense multiple access protocol with collision detection and arbitration on message priority (CSMA/CD+AMP) originally developed for use in automotive applications.The CAN data frame consists of an ID field, data field, Cyclic Redundancy Check (CRC), and other fields seen in Fig. 3 [5].

Research in the reverse engineering and translation of CAN messages is an active area. READ [6] is a novel algorithm that isolates counters and CRCs among other values to label signal types based on data frames in CAN traces. ACTT [7] leverages diagnostic information to parse CAN by breaking messages into tokens and then learning the translation from bits to vehicle function. LibreCAN [8] captures the bit-flip rate of messages and uses them along with sensor data from a smartphone to classify messages. CAN-D [9] extracts hidden signals (endianness and signedness) in CAN data using a four-step pipeline with machine learning, optimization, and heuristics to identify and correctly translate signals in CAN data to their numerical time series. The prior work focuses mainly on decoding the data frame by identifying the signal boundaries and correlating their changes with vehicle functions by monitoring them with a controlled vehicle. Our approach differs by abstracting the signals encoded in the data frames and instead focusing on the relationship between sets of frames in time to identify which messages correspond to the vehicle functions in a manner that does not require any special equipment and can be done without controlled vehicle experimentation or substantial automotive expertise.

III. Cluster-based Reverse Engineering

Raw CAN is unlabeled and requires proprietary information from the manufacturer to reverse engineer. Our approach to reverse engineer the relationship between CAN messages and vehicle functions with low domain expertise relies on unsupervised machine learning—clustering—to identify related groups of messages. Clustering machine learning techniques assume that instances of a particular class have data profiles that cluster into centroids. Each new data point can then be classified according to its distance from that centroid.

At a high level, our reverse engineering process involves reading CAN data, calculating the distances between all CAN IDs, merging nearest IDs together to form clusters, and using these clusters to classify unknown CAN IDs. The merging of clusters and termination of clustering depends on the specific clustering algorithm approach. When clustering completes, our hypothesis is that the merged clusters group the different CAN messages together by their respective functions.

We use the Euclidean distance metric by mapping each message to a point in the x-y plane based on its timestamp. The CAN ID and data field are extracted from CAN data frames to use as features, in addition to the receive timestamp. This information is available in all CAN log formats and are standard to all vehicles regardless of their manufacturer, make, or model. For all messages we set the x value to the origin, so the distance function reduces to the L^1 norm of $|y_1 - y_2|$, the timestamps of messages m_1 and m_2 respectively.

We generate a distance matrix for clustering by calculating the distance between the last transmissions of each ID. We smooth the distances by applying the arithmetic mean over the distances calculated in the final 100 transmissions of a captured CAN log. The distance matrix then shows the distance between each ID based on the averaged distance between their last few transmissions. This matrix is used as input to the clustering approaches. We use and compare the following 4 clustering algorithms for our approach to reverse engineering.

Agglomerative Hierarchical (AH) Clustering is based on k-means clustering. It ensures that two nearby points are placed in the same cluster. In our approach, this means that each CAN ID is assigned

as a single cluster and as we iterate through the IDs and their respective distance measurements, then CAN ID clusters are merged starting with the nearest neighbor. The advantage of this approach is that if certain CAN IDs have a known function, the functions of unknown CAN IDs can be determined by their clustering.

Genetic Algorithm (GA) Clustering uses randomized adaptive search heuristics that imitate the biological process of natural selection. Any GA starts with a set of randomly generated states of chromosomes known as the population. A fitness function that influences the next generation of states is calculated by measuring the quality of the clustering for each state. In our case, each state is a distance metric of a particular CAN ID from all other CAN IDs. We use a variant of the GA that requires the user to specify the number of clusters manually.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) defines a cluster as a maximal set of density-connected points. The algorithm requires two parameters: maximum radius of the neighborhood (Eps) and the minimum number of points in the Eps neighborhood of a point (MinPts). We varied Eps from $0-1$ with an increment of 0.05 and MinPts from $1-10$ with an increment of 1 to find a good parameter to cluster CAN IDs. We chose DBSCAN because it helps to discover structures in data that are difficult to observe.

Expectation Maximization Clustering with Gaussian Mixture Models (EM-GMM) is a type of soft clustering algorithm in which clusters may overlap. It starts by assuming that all the data points are Gaussian distributed. Each cluster then corresponds to a probability distribution, and EM-GMM tries to discover their parameters (mean and covariance). Although this algorithm requires prior knowledge of the number of clusters, we chose to use it because it can help to investigate if a particular CAN ID belongs with multiple vehicular functions.

IV. Evaluation

We implemented the 4 clustering approaches in Python with the numpy and scipy libraries, and with the PyClustering tool [10]. We evaluate our approach using 2 CAN data sets: simulated CAN data and real J1939.

	10	1	3	2	5	4	7	6	9	8
10	0.00	0.75	0.65	0.70	0.50	0.55	0.20	0.25	0.05	0.15
1	0.75	0.00	0.10	0.05	0.25	0.20	0.55	0.50	0.70	0.60
3	0.65	0.10	0.00	0.05	0.15	0.10	0.45	0.40	0.60	0.50
2	0.70	0.05	0.05	0.00	0.20	0.15	0.50	0.45	0.65	0.55
5	0.50	0.25	0.15	0.20	0.00	0.05	0.30	0.25	0.45	0.35
4	0.55	0.20	0.10	0.15	0.05	0.00	0.35	0.30	0.50	0.40
7	0.20	0.55	0.45	0.50	0.30	0.35	0.00	0.05	0.15	0.05
6	0.25	0.50	0.40	0.45	0.25	0.30	0.05	0.00	0.20	0.10
9	0.05	0.70	0.60	0.65	0.45	0.50	0.15	0.20	0.00	0.10
8	0.15	0.60	0.50	0.55	0.35	0.40	0.05	0.10	0.10	0.00

Fig. 4: Distance matrix for Simulated CAN Data.

A. *Simulated CAN Data*

We conducted a feasibility study using a simulated CAN data log with 10 different CAN IDs, simulating an acceleration mode and a braking mode. This data log had messages with different frequencies and message intervals to simulate real data. There were 5 IDs specifically related to acceleration and another 4 IDs related to braking, with 1 message that was transmitted during both modes. This data was used to verify and validate our hypothesis that clustering can be effective. Given that raw CAN data is dependent on the vehicle it was captured from, our simulated data gives us a baseline to understand the workings of our algorithms when applied towards raw captured CAN data. Simulated data was generated using Vector CANoe and has complete information about the relationship between messages and vehicle functions.

Figure 4 shows the distance matrix for this data. We passed this matrix into the AH clustering algorithm as well as a dendrogram creator. A dendrogram visualizes hierarchical clustering in a tree-like structure that records the sequences of merges of the leaves, in this case the CAN messages. The dendrogram (Fig. 5) is created from the leaves first to create the cluster tree. It shows that there are two clusters of the CAN IDs, which is expected since we simulated two separate driving modes of braking and accelerating. Our clustering algorithm confirmed the dendrogram results. CAN IDs 1-5 and 8 were designated as accelerating and were clustered into one group, and CAN IDs 6, 7, 9, and 10 were designated as braking and clustered into another group. The simulated data was used to confirm our hypothesis and approach before applying clustering to real CAN data.

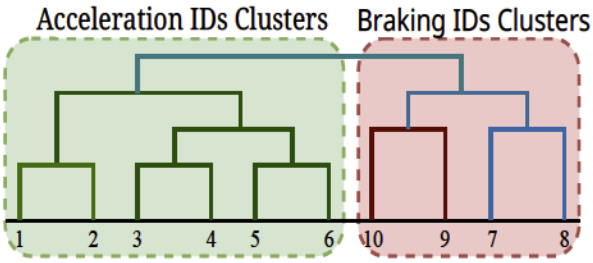

Fig. 5: Dendrogram for Simulated CAN Data. The two major groupings for the CAN IDs are easily distinguished visually.

B. Real J1939 Data

To address the chicken-and-egg problem of incomplete information about the ground truth that our clustering algorithms are trying to discover, we obtained CAN data that followed the J1939 standard to examine the performance of our approach. The Society of Automotive Engineers standard SAE J1939 is the vehicle bus standard based on CAN that has been widely adopted by diesel engine manufacturers for use in large tractors and trucks [11]. This standard defines that all J1939 packets, except for the request packet, should contain eight bytes of data and a standard header which contains a Parameter Group Number (PGN), that is embedded in the message's 29-bit identifier. A PGN identifies a message's function and associated data. J1939 attempts to define standard PGNs to encompass a wide range of automotive and other vehicle purposes. PGNs define the message functions so we validate the clustering algorithm results by using the standard as a basis for ground truth.

The J1939 data set was captured from a John Deere tractor as it performed tillage operations at different sites. This data set is raw, has no labels, and contains over 300 CAN IDs. We created the distance matrix for this data and passed it to the 4 clustering algorithms.

To measure the performance of the clustering algorithms, we manually generated sets of related PGNs to define the IDs that belong to similar vehicular functions. We filtered out some PGNs that are not well-defined and omit them from calculation of performance. Performance is calculated by comparing the manually labeled ground truth and each algorithm's generated clusters using the Fowlkes-Mallows score (FM-Score) [12]. FM-Score is an extrinsic clustering evaluation metric that evaluates the similarities between two clusters, i.e., the ground truth and each algorithm's output. Table I summarizes the results of the FM-Scores for each of the four clustering approaches we investigated.

TABLE I: Cluster FM-Scores with J1939 Data.

Clustering Algorithm	FM-Score
Agglomerative Hierarchical (AH)	71.72%
Genetic Algorithm (GA)	41.03%
DBSCAN	54.90%
EM-GMM	71.82%

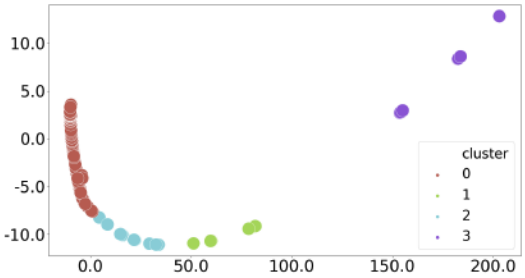

Fig. 6: PCA of AH Clustering

The AH and EM-GMM algorithms performed the best, and we focus further on them to understand the benefits of clustering. With this data, which is much richer, the dendrogram is less useful, so we visualized the high-dimensional clusters using principal component analysis (PCA) to help guide our parameter tuning. An example of the case using AH clustering is shown in Fig. 6. The AH algorithm returned four clusters (0-3) as shown in Table II. The largest cluster contained all the driving critical functions, such as braking and engine controller. The other three clusters were much smaller and contained proprietary IDs, memory accesses, and data transfer IDs in separate clusters. The results confirm that related CAN messages are clustered together by unsupervised learning algorithms. Given that J1939 CAN data has known functions, we were able to confirm that CAN IDs with related functions cluster together.

V. Conclusion

In this paper, we have motivated the need for vehicle security in smart cities and evaluate how

TABLE II: AH Clusters found in J1939 data.

Cluster Label	CAN Function
0	Driving (Engine and Brake)
1	Proprietary
2	Memory Access
3	Data Transfer

well unsupervised learning may characterize the externally visible actions of vehicles with messages transmitted in the CAN bus. The experimental results show that unsupervised clustering methods are able to classify, identify, and label specific CAN IDs by their respective functions. These methods only require timestamped CAN logs—they do not require insider knowledge or previously reverse-engineered data. Our approach applies to all vehicles but we could only validate the algorithm on J1939 CAN protocol because the DBC for other vehicles was not accessible. We could easily integrate this approach to any vehicle that utilizes CAN related protocols. Such low-effort capabilities, combined with the data-rich smart city ecosystem, may lead to greater understanding of how to secure connected vehicles in the future.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. CNS 1646317, CNS 1645987, and CNS 2011620.

REFERENCES

[1] T. Alladi, V. Chamola, B. Sikdar, and K. R. Choo, "Consumer iot: Security vulnerability case studies and solutions," *IEEE Consumer Electronics Magazine*, vol. 9, no. 2, pp. 17–25, 2020.

[2] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," in *2010 IEEE Symposium on Security and Privacy*, May 2010, pp. 447–462.

[3] H. Olufowobi, C. Young, J. Zambreno, and G. Bloom, "Saiducant: Specification-based automotive intrusion detection using controller area network (can) timing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 1484–1494, 2019.

[4] C. Young, J. Zambreno, H. Olufowobi, and G. Bloom, "Survey of automotive controller area network intrusion detection systems," *IEEE Design & Test*, vol. 36, no. 6, pp. 48–55, 2019.

[5] M. D. Natale, H. Zeng, P. Giusto, and A. Ghosal, *Understanding and Using the Controller Area Network Communication Protocol: Theory and Practice*. Springer Publishing Company, Incorporated, 2012.

[6] M. Marchetti and D. Stabili, "Read: Reverse engineering of automotive data frames," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1083–1097, 2018.

[7] M. Verma, R. Bridges, and S. Hollifield, "Actt: Automotive can tokenization and translation," in *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2018, pp. 278–283.

[8] M. D. Pesé, T. Stacer, C. A. Campos, E. Newberry, D. Chen, and K. G. Shin, "Librecan: Automated can message translator," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2283–2300.

[9] M. E. Verma, R. A. Bridges, J. J. Sosnowski, S. C. Hollifield, and M. D. Iannacone, "Can-d: A modular four-step pipeline for comprehensively decoding controller area network data," *arXiv preprint arXiv:2006.05993*, 2020.

[10] A. Novikov, "PyClustering: Data mining library," *Journal of Open Source Software*, vol. 4, no. 36, p. 1230, apr 2019.

[11] S. Mukherjee, J. C. Van Etten, N. R. Samyukta, J. Walker, I. Ray, and I. Ray, "Truckstm: Runtime realization of operational state transitions for medium and heavy duty vehicles," *ACM Trans. Cyber-Phys. Syst.*, vol. 4, no. 1, Nov. 2019.

[12] E. B. Fowlkes and C. L. Mallows, "A method for comparing two hierarchical clusterings," *Journal of the American Statistical Association*, vol. 78, no. 383, pp. 553–569, 1983.

ABOUT THE AUTHORS

Uchenna Ezeobi is a PhD student in the Department of Computer Science at University of Colorado Colorado Springs. Contact him at uezeobi@uccs.edu

Habeeb Olufowobi is an Assistant Professor in the Department of Computer Science and Engineering at University of Texas Arlington. Contact him at habeeb.olufowobi@uta.edu

Clinton Young completed his PhD in the Department of Electrical and Computer Engineering at Iowa State University. Contact him at cwyoung@iastate.edu

Joseph Zambreno is a Professor in the Department of Electrical and Computer Engineering at Iowa State Univerity. Contact him at zambreno@iastate.edu

Gedare Bloom is an Assistant Professor in the Department of Computer Science at University of Colorado Colorado Springs. Contact him at gbloom@uccs.edu