Fine-grained Graph-based Anomaly Detection on Vehicle Controller Area Networks

Isaiah J. King[†*], Benjamin Bowman[†], H. Howie Huang[†*]
[†]Cybermonic LLC, McLean, VA USA, benjamin@cybermonic.com
[*]The George Washington University, Washington, DC USA, {iking5, howie}@gwu.edu

***Abstract*—Electronic components in vehicles communicate with one another by broadcasting messages over the controller area network (CAN) bus. The CAN message protocol is notoriously insecure, lacking both encryption and authentication for performance reasons. Vehicle manufacturers instead opt for "security through obscurity" and try to keep the meanings of CAN messages industry secrets. This approach has led to the discovery of several alarming, and unaddressed vulnerabilities. For this reason, it is imperative to develop a security monitoring system for the CAN bus. However, any such intrusion detection system is limited by severe memory constraints–in-vehicle ECUs rarely have more than 1MB of RAM. In this work, we explore the potential for lightweight graph kernel-based intrusion detection systems that work in conjunction with byte analysis of individual messages. Our approach extends the state-of-the-art in this field, which only classifies batches of messages as malicious or benign, rather than performing fine-grained anomaly detection. We analyze the precedence graph formed by CAN message ordering in conjunction with the bytes those messages contain to create a high-performance, low-memory anomaly detector. Our analysis revealed that this approach can detect a wide variety of attack types in both moving and stationary vehicles. We demonstrated that our method performs more precisely than prior works in the same field while requiring less than 100KB of memory.**

***Index Terms*—Automotive electronics, directed graphs, graph kernels, network security, vehicle safety**

I. Introduction

Increasingly, auto manufacturers are filling vehicles with computers: machines to measure RPM, monitor engine heat, track fuel usage, and even roll down your windows. These computers need some way to communicate. This is the purpose of the controller area network (CAN) bus. The CAN bus allows each electronic control unit (ECU) to send messages to any other microcontroller on the network [7]. By using a centralized bus, and broadcasting each message, auto manufacturers can minimize cost, as there is no need for a router, and very little wiring is used to connect each component. Though the CAN protocol is used extensively in heavy machinery, consumer vehicles, and airplanes, it is extremely vulnerable to attacks. Because the CAN protocol is designed for speed and simplicity, there is no encryption, authentication, or network segmentation [47]. While this makes communication extremely efficient, it has some alarming security consequences. For example, an attacker with access to the CAN bus could remotely accelerate a vehicle to 200km/hr [45], remotely track the vehicle's location [43], or even deploy its airbags [1], [5].

Many have suggested changing the standard to include encryption, or authentication to prevent such attacks [8], [30], [41], but this still leaves older vehicles vulnerable and comes at a significant cost. To address this, there has been growing interest in intrusion detection systems (IDS) to monitor existing CAN buses for malicious activity. There are many machine learning-based approaches to solve this problem, but these approaches utilize compute and memory-intensive neural networks [17], [18], [29], [33], [35], [38] and would be difficult to implement in the memory-constrained real world. While complex ML models can filter malicious messages with near-perfect precision, it is not a valid assumption that every CAN bus will have a dedicated GPU for those models to live on. The ECUs identified by Wolf [44] as widely adapted in commercial vehicles have at the high, expensive end, just 1.6MB of memory [12]. Given this constraint, it is unrealistic to suggest powerful deep neural networks as a solution to this problem. With this in mind, we turn to lightweight machine learning approaches.

The current state-of-the-art in this regard are graph kernel methods [13], [28]. Graph kernel-based IDS [34] works by representing a system as a graph. Then, by using a graph kernel, or a function that maps from the graph domain to the vector domain, they compare the vector representation of the graph to some baseline for normal behavior. A good graph kernel function will produce vectors such that similar graphs will be close to each other in Euclidian space. This approach posits that given enough samples of a system's normal behavior, the manifold that bounds them will contain all normal activity. Any graphs that are projected to points outside of this space are said to be anomalous. This approach has seen success for intrusion detection systems and anomaly detection on traditional computer networks [2], [10], [19], [20], [23], now we extend it to inner vehicle networks.

Messages on the CAN bus, however, are very different to those found on traditional computer networks. One key difference is that every message is broadcast to every ECU [14], so a graph representing inter-ECU communication would be a complete graph. Though every message is broadcast, only certain ECUs are interested in messages from specific CAN IDs. Though it may seem like this means inter-ECU communication graphs could be meaningful, the knowledge of which ECUs react to messages from which CAN IDs is a closely guarded secret [37]. The sender, receiver, and meaning of the contents of CAN messages is proprietary, so constructing a graph in this manner is unreasonable. Instead, graph kernel-based approaches to CAN IDS represent communication via prece-

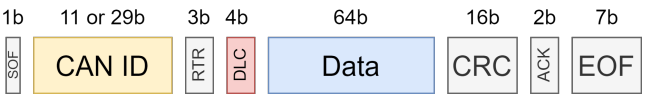
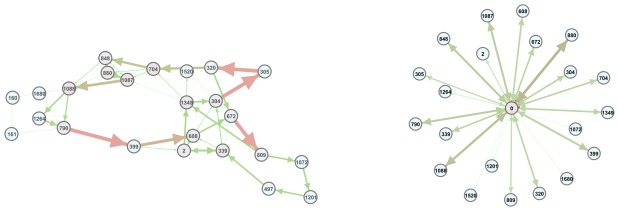


Fig. 1: A CAN frame according to CAN 2.0 protocol. Under CAN 2.0A, the CAN ID is 11b; under CAN 2.0B, CAN IDs are 29b.



(a) Normal operations. (b) Flooding attack.

Fig. 2: Two example precedence graphs generated based on the Car Hacking Dataset for Intrusion Detection.

dence graphs [6]. The precedence graph is a data structure that captures the precedence, or ordering, of CAN IDs over discrete time intervals. These graphs are so effective because they often approximate which components are communicating with each other. For example, in the J1939 standard [31], many CAN messages are requests for information from other ECUs. The other ECUs quickly respond with an answer after receiving the initial message, manifesting in an edge between those CAN IDs in the precedence graph. We will analyze the precedence graphs constructed during malicious and benign periods, and we will show how doing this allows us to quickly detect when the CAN bus is under attack.

Though prior works on precedence graph analysis are effective at detecting malicious periods of time in CAN data, they still the lack fine-grained level of alerts required to be useful as a real IDS. Because graph kernel-based approaches label *periods of time* as malicious, any CAN messages sent while the vehicle was under attack would be misclassified as malicious as well. Assuming the goal is to intercept and block potentially malicious messages, graph kernel-based approaches would effectively disable the vehicle during the attack period, which hardly mitigates the threat. For an IDS to be useful, it is not sufficient for it to simply detect that the CAN bus is under attack. An effective IDS would tell us which specific messages are malicious, giving us the potential to mitigate an attack.

In this work, we present and evaluate our novel, fine-grained, unsupervised detection method that can be applied to *individual* CAN bus messages[1]. Our approach combines precedence graph analysis with a simple byte-threshold technique to detect anomalous messages within graphs that appear anomalous. We will show how our surprisingly simple approach produces improved results on several benchmark datasets while remaining lightweight and memory-efficient enough to run on a real-world ECU. We will show that our approach achieves state-of-the-art results on a variety of datasets and across several different attack types.

II. Background

CAN Messages: Vehicles, both commercial and industrial, are increasingly reliant on embedded computers called electronic control units (ECUs). These ECUs communicate using one or more CAN busses. They broadcast messages to every component at the same time. These messages are formatted using the CAN standard [14]. We illustrate a simplified version of this standard in Figure 1. In this work, we only consider the CAN ID field, and the data field. The CAN ID is an 11 or 29b identifier for what message is being sent. Importantly, these identifiers map onto the set of ECUs, meaning each CAN ID originates from exactly one ECU, but each ECU may produce CAN messages with several different IDs. Each message can contain up to 8 bytes of data.

Precedence Graphs: A graph is defined as a set of discrete objects, called nodes, $\mathcal{V}$, and a set of relationships between those objects called edges, $\mathcal{E} \subseteq \mathcal{V} \otimes \mathcal{V}$, where $\otimes$ represents the Cartesian product. A weighted graph is a graph where each edge has a (positive) weight associated with it, $f : \mathcal{E} \to \mathbb{R}^+$. Because CAN messages have no obvious intended recipient, it would be difficult to represent CAN communications as a communication graph, as is often done in graph-based IDS [3] As a result, prior works [13] and [24] model CAN data as a *precedence graph* [6]. These data structures model the ordering of events–in this case, the order in which CAN messages are sent. Each unique CAN ID is a node, and observing one ID after another creates a directed edge between them.

We illustrate two sample precedence graphs in Figure 2. The subfigure 2(a) shows how during normal operations, certain nodes have edges with very high weight (how often they are observed in a given timeframe). In the DBC for the J1939 CAN standard [31], they write that many messages are requests for information from other ECUs. The ECUs being queried will then broadcast the information being asked of them. This is likely what causes certain edges to be so heavily weighted in the precedence graph. On the other hand, The subfigure 2(b) illustrates the precedence graph formed during a flooding attack. This attack involves filling the CAN bus with empty frames to disrupt normal operations, causing denial of service. Now, rather than a chain structure of CAN IDs messaging and responding to each other, we observe a star topology with a central hub at CAN ID `0x00`. This is because the CAN frames injected during the attack were empty, meaning the address is 0. Visually, distinguishing between these two graphs is easy, but to do so numerically, we turn to graph kernels.

Graph Kernel Methods: A graph kernel is a function that maps from a graph to a vector, $\Phi : \mathcal{G} \to \mathbb{R}^d$. Ideally $d \ll |\mathcal{E}|$, and Φ encodes key information about the graph, such that for small values of ε, $\|\Phi(\mathcal{G}_1) - \Phi(\mathcal{G}_2)\| - \varepsilon = 0$ implies that the graphs are similar in some way. In traditional cybersecurity, benign graphs are sampled from the data, and the IDS models

[1]Source code available at https://www.github.com/cybermonic/CAN-IDS

attempt to find the boundary on the space $\mathbb{R}^d$ that encloses the space of all normal activity [10], [20], [23]. In the world of CAN analysis, we have only found one method which applies this technique for anomaly detection: G-IDCS [28]. It is the only graph kernel technique, and as far as we can tell, the best approach to CAN anomaly detection available.

As we see in G-IDCS's success, as well as other methods in host- and network-based IDS, this method of unsupervised anomaly detection can work quite well. Moreover, when training is complete, these methods are fast enough for real-time use, and–as we will show–memory efficient enough to fit into a real-world ECU. However, graph kernel-based IDS has a crucial limitation: these approaches classify full graphs as either malicious or benign. They are not fine-grained. Unless messages are individually classified as malicious or benign, these coarse-grained IDS approaches will filter many more benign messages than malicious ones, making them unrealistic for real-world use.

III. Method

In this section, we will discuss how we convert from a stream of CAN messages to a precedence graph. Then, we will describe the graph kernel we used to analyze the precedence graphs and a simple byte thresholding method we used to perform edge-level anomaly detection. Finally, we will discuss how we combined the graph kernel method with the edge-level detection method. This forms an ensemble model that fine-tunes the powerful graph kernel methods.

A. Precedence Graph Encoding

Given a list of all CAN messages that were transmitted throughout a mission, converting them into a precedence graph is very simple. Let $\mathbf{C}$ represent the $|\mathcal{M}| \times 1$ vector of every CAN ID that appeared in the stream of messages, $\mathcal{M}$. We can then form an edge list by concatenating $\mathbf{C}$ with itself, offset by one:

$$\mathcal{E} = \begin{bmatrix} \mathbf{C}_{0:|M|-1} \\ \mathbf{C}_{1:|M|} \end{bmatrix} \tag{1}$$

the output is the $2 \times |M| - 1$ dimensional edge list.

This matrix can then be subdivided into batches of N edges. Like prior work [28], we also use batches of size 200. When replicating the graph kernel from G-IDCS, we need to identify three features about each batch of 200 edges: the elapsed time it took to produce that many CAN messages, the maximum degree in the graph, and number of unique edges. Calculating the elapsed time is trivial, and is inferred from the timestamps of the CAN messages.

Calculating the number of unique edges, and highest degree node requires iterating over each batch twice. During a single pass, using two hashmaps, the algorithm counts the pairs of $\langle src, dst \rangle$ it observes, and the individual nodes src and dst. Because we define node degree as the sum of inbound edges and outbound edges, counting node occurrences in the top and bottom rows of the edge index is sufficient. Then, the map of counts must be iterated over again to find the maximum value.

TABLE I: Dataset Metadata

		Edges	Malicious Edges	% Malicious
CHD	DoS	3,665,770	587,521	16.03
	Fuzzy	3,838,859	491,847	12.81
	Gear	4,443,141	597,252	13.44
	RPM	4,621,701	654,897	14.17
ADCD	Stationary	4,818,638	1,171,665	24.32
	Driving	3,875,860	772,576	19.93

B. Anomaly Detection

To detect anomalies on the byte level, we use a simple thresholding approach. Let $\mathbf{Z}$ denote the encoding of a period of normal activity generated using either of the graph kernels we described in the previous section. We then calculate the maximum, and minimum values across each row in the matrix. This results in two new vectors, $\mathbf{z}_{max}$ and $\mathbf{z}_{min}$. These two vectors are the only parameters required by the model.

During the inference stage, as CAN messages are streamed in, they are converted into a new precedence graph, $\mathcal{G}'$ using the previously described method. We say that a graph is anomalous if

$$\exists i : \Big(\Phi(\mathcal{G}')_i > (1+\epsilon_g)\mathbf{z}_{max,i}\Big) \vee \Big(\Phi(\mathcal{G}')_i < (1-\epsilon_g)\mathbf{z}_{min,i}\Big). \tag{2}$$

Here, Φ is the graph kernel function, and ϵ_g is a threshold value, 0.03 in all experiments we conducted. Of course, this will only detect batches of N CAN messages as anomalous.

For fine-grained anomaly detection, we implement another thresholding method. For each CAN message ID observed during the period of normal activity, we track the maximum and minimum values of each byte in the data frame of each message they emit. These values are stored in hashmaps `Max` and `Min`. Then, at inference time, we say a message $\mathcal{M}_t$ from CAN ID i as anomalous if it appeared in an anomalous graph, and if

$$\mathcal{M}_t > (1+\epsilon_b)\texttt{Max}[i] \vee \mathcal{M}_t < (1-\epsilon_b)\texttt{Min}[i] \tag{3}$$

where ϵ_b is another thresholding parameter, also set to 0.03. We will show how this method combines the best parts of each anomaly detection method: graph kernels' high recall, and byte thresholds' high precision.

IV. Experiments

A. Evaluation Datasets

We evaluate the anomaly detection methods on two datasets: the Car-Hacking Dataset (CHD) [32], and the Attack & Defense Challenge Dataset (ADCD) [16]. The CHD contains CAN messages collected from a real vehicle undergoing 4 different attacks. We used benign periods from the attack data to train our models during the experiments. The ADCD contains data from four different attacks: fuzzing, DoS, spoofing, and replay attacks. The files for this dataset are split into categories based on if the vehicle undergoing the attacks was stationary or moving. Additional details about the contents of each dataset are available in Table I

B. Experiment Results

In our experiments, We evaluate IDS models on a per-message basis. This means that these detection approaches could be implemented as potential automated countermeasures. If we assume that there is some capability on the network to inspect and filter messages, then any message that is flagged as malicious could be dropped, thereby mitigating the attack. For these experiments, we evaluated G-IDCS [28], the byte thresholding method, the DAGA model [36], and our proposed hybrid approach. G-IDCS is the base graph kernel we use in our hybrid model; however, it can only classify full graphs as malicious or benign, so we expect to observe a low precision score when it is evaluated on its own. DAGA interprets sequences of CAN IDs as n-grams. It stores every n-gram of sequential CAN IDs it has seen in benign data, then at runtime, it raises an alert if the sequence of a new message and the previous $n-1$ messages has never been seen before. This approach is notable in that it is also memory-conscious and provably can be deployed on a real-world vehicle microprocessor due to its small memory footprint. However, in the original work, it was primarily tested on replay attacks.

Tables II and III show the performance of the edge-level anomaly detectors on the CHD and ADCD datasets, respectively. We observe poor performance in G-IDCS's precision; it has a high recall, as it can quickly tell that several messages in the aggregate are anomalous, but it is unable to determine which ones make it so. This supports our claim that most groups of messages inspected by the graph kernel methods contain mostly benign messages, as this points to an increase in false positives. In particular, the spoofing attacks have the lowest precision when evaluated this way as they are slower and stealthier than the other attacks. These results are notable compared to the evaluation done in the original G-IDCS paper: there, it was evaluated only on groups of N messages and achieved scores > 0.99 across every metric. When evaluated on the message level, there is a sharp decline in the score. Surprisingly, the DAGA model performed about equally, or slightly worse than G-IDCS did across both datasets. It could be that it was too specialized for replay attacks to generalize to different attack types, or just that there was not sufficient training data for it to observe every possible benign n-tuple.

Especially in the CHD dataset, the simple byte threshold method was able to achieve 1.0 recall on every attack. This makes sense for the fuzzing and DoS attacks, as they flood the can bus with `0x00` bytes or random values respectively, so they are easy to capture. In the ADCD dataset, because attacks are mixed together, the Byte Threshold detector does not perform as well.

To take advantage of the high recall of the Byte Threshold, and the high precision of G-IDCS (when evaluated at the graph level), we combine the two approaches into a hybrid classifier. This approach only alerts if the G-IDCS classifier labels a period of messages anomalous, *and* there is an anomalous byte in the message. Though this approach is simple, it outperforms both of its component models in nearly every metric. By using G-IDCS as a filter for what to send to the more alert-prone Byte Threshold, we get the best of both models. The graph kernel very precisely senses which periods of time are anomalous, and then the Byte Threshold analyzes which messages cause this. When the two are combined into the hybrid model, important metrics like F1 reach the high 90s.

TABLE II: CHD Message-level Anomaly Detection

		G-IDCS	Byte Thresh	DAGA[2]	Hybrid
DoS	Accuracy	0.8522	**0.9984**	0.8578	0.9983
	Precision	0.5203	**0.9903**	0.5300	**0.9903**
	Recall	0.9992	**1.0000**	0.9921	0.9992
	F1	0.6843	**0.9951**	0.6909	0.9948
Fuzzy	Accuracy	0.7828	0.9819	0.8085	**0.9924**
	Precision	0.3708	0.8764	0.3903	**0.9458**
	Recall	0.9980	**1.0000**	0.8790	0.9980
	F1	0.5407	0.9341	0.5405	**0.9712**
Gear	Accuracy	0.6987	0.9642	0.7789	**0.9838**
	Precision	0.3085	0.7895	0.3346	**0.8924**
	Recall	0.9997	**1.0000**	0.6526	0.9997
	F1	0.4715	0.8824	0.4424	**0.9430**
RPM	Accuracy	0.6852	0.9636	0.7012	**0.9825**
	Precision	0.3104	0.7955	0.2869	**0.8903**
	Recall	0.9996	**1.0000**	0.7467	0.9996
	F1	0.4737	0.8861	0.4146	**0.9418**

TABLE III: ADCD Message-level Anomaly Detection

		G-IDCS	Byte Thresh	DAGA[3]	Hybrid
Stationary	Accuracy	0.6240	0.5712	0.5150	**0.8120**
	Precision	0.2465	0.1838	0.1519	**0.3534**
	Recall	**0.9983**	0.8189	0.7234	0.8118
	F1	0.3954	0.3002	0.2510	**0.4924**
Moving	Accuracy	0.5951	0.6581	0.5115	**0.9967**
	Precision	0.1955	0.1929	0.1256	**1.0000**
	Recall	**0.9686**	0.8133	0.6919	0.9651
	F1	0.3254	0.3119	0.2125	**0.9823**

A notable exception to this is the data from the stationary vehicle in the ADCD dataset. Though the hybrid approach is still the best-performing one, it still lacks precision. This is likely because the benign stationary vehicle data does not encompass the entire benign space of possible CAN messages, but this requires further analysis.

V. Efficiency Study

Prior work [36] found that in realistic automotive ECUs, there is often less than 1MB of flash memory. Therefore, to support our claim that this approach is suitable for real-world use, we evaluate the runtime and memory usage of a C++ implementation of our hybrid approach. We used the `std::unordered_map<int, char*>` as the data structure for the max and min byte lookup tables, and represent the precedence graph as a single `int` array of CAN IDs. Other than using the `boost::hash_combine` function to define

[2]Using optimal n-grams lengths 3,4,4, and 5.

[3]Using optimal n-gram length 3 for both datasets

the hash function used by the map, the code only uses the standard C++17 library. Using the Valgrind memory profiling tool [27], we found that the peak memory use was 95.4KB, which is well within the constraints of mid-range ECUs.

The asymptotic bound on memory use is constrained by the size of the precedence graph. Each graph requires N CAN IDs, and 8 bytes of data to be stored. The maps from CAN IDs to byte max and min thresholds require $|C| \times 8$ bytes, where $|C|$ is the number of unique CAN IDs. In our experiments, and in most real-world use cases [9], $|C| \ll N$. Thus, the worst-case memory use of this approach is $\mathcal{O}(N)$.

In addition to memory usage, CAN bus security is constrained by runtime. Using the graph kernel requires iterating through the list of CAN IDs one time. It counts unique edges using a `std::unordered_set<tuple<int, int>>`, which has $\mathcal{O}(1)$ access time. It finds the maximum degree using an unordered map to count CAN ID occurrences, then iterates through the map again to find the largest value. In the worst case, each can ID was unique, and this process would require $2N$ steps. Thus, the asymptotic upper bound is also linear at $\mathcal{O}(N)$. In wall-clock time, we found that processing the CAN streams using the G-IDCS graph kernel and analyzing the bytes of potentially malicious messages from 10,000 graphs took a total of 0.1109s, and 0.0229s respectively. In real-time, it took 1,079s for the CAN bus to have enough activity to build 10,000 graphs. Thus, we can conclusively say our approach works faster than real time.

VI. Related Work

Automotive Security encompasses far more than just CAN security. In addition to CAN, vehicles use local interconnection networks, media oriented system transport, and automotive ethernet [4], not to mention the connections to external USB and bluetooth devices. Automotive ethernet, though not yet widely adopted, has many proposed security measures. For example, the higher throughput afforded by its protocol allows for easier encryption [46], [22] than CAN. It also allows for more memory intensive solutions such as convolutional neural network-based IDS [15].

CAN IDS can largely be broken into four classifications: signal-, physical-, frequency-, or payload-based [42]. Our approach falls under the payload- and frequency-based categories of this taxonomy. Payload-based techniques process the data section analyze the signals being sent by various ECUs without explicitly decoding them. CANET [11], for example, uses a multi-LSTM architecture to process the payloads of CAN messages from specific IDs using unsupervised learning. Supervised learning approaches like [17], [38] have also shown great promise in this field, though we are primarily interested in unsupervised anomaly detection. Prior work [39] trains several one-class classifiers to find the hyperspheres that describe the normal distribution of bytes generated by each CAN ID. Other approaches use more traditional ML approaches such as ARIMA [40] or hidden Markov models [26] to fit CAN message distributions. The byte analysis of our approach loosely fits into this category, though it is not as advanced as some of these techniques.

Frequency-based approaches find anomalies in message arrival times [21], or in message ordering [25]. Works like G-IDCS [33] and Islam et al. [13] use message arrival order to form precedence graphs.

VII. Conclusion

There is great potential for future work in implementing this system on a real-world vehicle. Filtering CAN messages before they are allowed onto the bus using simple thresholding rules and by analyzing them in the context of the precedence graph appears to be a sufficient method of threat mitigation. Additionally, with expert knowledge, and insight into the meanings of CAN messages, expected ranges of bytes in CAN messages could be defined explicitly and used for even better message-level anomaly detection.

In this work, we evaluated IDS approaches for the CAN bus on individual CAN messages. We found that kernel models have deceptively high metrics when evaluated at the batch level; when evaluated on a per-message basis, their precision drops below an acceptable level. By ensembling G-IDCS with a simple byte threshold anomaly detector, we were able to improve its precision and recall to a point where it could be feasible for real-world use. We demonstrated that this approach requires very little compute or memory and could be implemented in a real-world vehicle.

Acknowledgments

This work was funded in part by US Army SBIR Grant W5170123C0142. The views expressed are those of the authors and do not reflect the official policy or position of the US Army.

References

[1] "CVE-2017-14937," Available from NIST vulnerability database (NVD), 2017. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2017-14937

[2] B. Bhattarai and H. H. Huang, "Prov2vec: Learning provenance graph representation for anomaly detection in computer systems," in *Proceedings of the 19th International Conference on Availability, Reliability and Security*, 2024, pp. 1–14.

[3] B. Bowman, C. Laprade, Y. Ji, and H. H. Huang, "Detecting lateral movement in enterprise computer networks with unsupervised graph AI," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, pp. 257–268.

[4] A. B. C. Douss, R. Abassi, and D. Sauveron, "State-of-the-art survey of in-vehicle protocols and automotive ethernet security and vulnerabilities," *Mathematical Biosciences and Engineering*, vol. 20, no. 9, pp. 17 057–17 095, 2023.

[5] J. Dürrwang, J. Braun, M. Rumez, and R. Kriesten, "Security evaluation of an airbag-ecu by reusing threat modeling artefacts," in *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2017, pp. 37–43.

[6] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems (5th Edition)*. USA: Addison-Wesley Longman Publishing Co., Inc., 2006, ch. 17.

[7] M. Farsi, K. Ratcliff, and M. Barbosa, "An overview of controller area network," *Computing & Control Engineering Journal*, vol. 10, no. 3, pp. 113–120, 1999.

[8] B. Groza and P.-S. Murvay, "Security solutions for the controller area network: Bringing authentication to in-vehicle networks," *IEEE Vehicular Technology Magazine*, vol. 13, no. 1, pp. 40–47, 2018.

[9] C. Hammerschmidt, "Number of automotive ECUs continues to rise," *eeNews Automotive*, 2019. [Online]. Available: https://www.eenewseurope.com/en/number-of-automotive-ecus-continues-to-rise/

[10] X. Han, T. Pasquier, A. Bates, J. Mickens, and M. Seltzer, "Unicorn: Runtime provenance-based detector for advanced persistent threats," *Network and Distributed Systems Security Symposium (NDSS)*, 2020.

[11] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, "Canet: An unsupervised intrusion detection system for high dimensional can bus data," *Ieee Access*, vol. 8, pp. 58 194–58 205, 2020.

[12] Infineon Technologies AG, "Infineon XC2000 family brochure," accessed Aug. 2, 2024. [Online]. Available: https://www.infineon.com/dgdl/Infineon%20-%20Brochure%20-%20XC2000%20Brochure.pdf?fileId=db3a304325305e6d0125366478d438e6

[13] R. Islam, R. U. D. Refat, S. M. Yerram, and H. Malik, "Graph-based intrusion detection system for controller area networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 1727–1736, 2020.

[14] "Road vehicles — Controller area network (CAN)," International Organization for Standardization, Geneva, CH, Standard, 2024.

[15] S. Jeong, B. Jeon, B. Chung, and H. K. Kim, "Convolutional neural network-based intrusion detection system for avtp streams in automotive ethernet-based networks," *Vehicular Communications*, vol. 29, p. 100338, 2021.

[16] H. Kang, B. I. Kwak, Y. H. Lee, H. Lee, H. Lee, and H. K. Kim, "Car hacking: Attack defense challenge 2020 dataset," 2021. [Online]. Available: https://dx.doi.org/10.21227/qvr7-n418

[17] M.-J. Kang and J.-W. Kang, "Intrusion detection system using deep neural network for in-vehicle network security," *PloS one*, vol. 11, no. 6, p. e0155781, 2016.

[18] T. Kim, J. Kim, and I. You, "An anomaly detection method based on multiple lstm-autoencoder models for in-vehicle network," *Electronics*, vol. 12, no. 17, p. 3543, 2023.

[19] I. J. King and H. H. Huang, "Euler: Detecting network lateral movement via scalable temporal link prediction," *ACM Transactions on Privacy and Security*, vol. 26, no. 3, pp. 1–36, 2023.

[20] I. J. King, X. Shu, J. Jang, K. Eykholt, T. Lee, and H. H. Huang, "Edgetorrent: Real-time temporal graph representations for intrusion detection," in *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*, 2023, pp. 77–91.

[21] H. Lee, S. H. Jeong, and H. K. Kim, "Otids: A novel intrusion detection system for in-vehicle network by using remote frame," in *2017 15th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2017, pp. 57–5709.

[22] J.-M. Li, Y.-J. W. Shuo-Fu, and Y.-N. Xu, "High-efficiency encryption and authentication network security for automotive ethernet," *International Journal of Modeling and Optimization*, vol. 12, no. 2, pp. 36–43, 2022.

[23] E. Manzoor, S. M. Milajerdi, and L. Akoglu, "Fast memory-efficient anomaly detection in streaming heterogeneous graphs," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1035–1044.

[24] M. Marchetti and D. Stabili, "Anomaly detection of can bus messages through analysis of id sequences," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 1577–1583.

[25] M. R. Moore, R. A. Bridges, F. L. Combs, M. S. Starr, and S. J. Prowell, "Modeling inter-signal arrival times for accurate detection of can bus signal injection attacks: a data-driven approach to in-vehicle intrusion detection," in *Proceedings of the 12th annual conference on cyber and information security research*, 2017, pp. 1–4.

[26] S. N. Narayanan, S. Mittal, and A. Joshi, "Obd_securealert: An anomaly detection system for vehicles," in *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2016, pp. 1–6.

[27] N. Nethercote and J. Seward, "Valgrind: a framework for heavyweight dynamic binary instrumentation," *ACM Sigplan notices*, vol. 42, no. 6, pp. 89–100, 2007.

[28] S. B. Park, H. J. Jo, and D. H. Lee, "G-idcs: Graph-based intrusion detection and classification system for can protocol," *IEEE Access*, 2023.

[29] K. Pawelec, R. A. Bridges, and F. L. Combs, "Towards a can ids based on a neural network data field predictor," in *Proceedings of the ACM workshop on automotive cybersecurity*, 2019, pp. 31–34.

[30] M. D. Pesé, J. W. Schauer, J. Li, and K. G. Shin, "S2-can: Sufficiently secure controller area network," in *Proceedings of the 37th Annual Computer Security Applications Conference*, 2021, pp. 425–438.

[31] B. Prasad, J.-J. Tang, and S.-J. Luo, "Design and implementation of sae j1939 vehicle diagnostics system," in *2019 IEEE International Conference on Computation, Communication and Engineering (ICCCE)*. IEEE, 2019, pp. 71–74.

[32] E. Seo, H. M. Song, and H. K. Kim, "Gids: Gan based intrusion detection system for in-vehicle network," in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, Aug 2018, pp. 1–6.

[33] ——, "Gids: Gan based intrusion detection system for in-vehicle network," in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2018, pp. 1–6.

[34] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels." *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.

[35] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Vehicular Communications*, vol. 21, p. 100198, 2020.

[36] D. Stabili, L. Ferretti, M. Andreolini, and M. Marchetti, "Daga: Detecting attacks to in-vehicle networks via n-gram analysis," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 11, pp. 11 540–11 554, 2022.

[37] J. Staggs, "How to hack your mini cooper: reverse engineering can messages on passenger automobiles," *Institute for Information Security*, 2013.

[38] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with long short-term memory networks," in *2016 IEEE international conference on data science and advanced analytics (DSAA)*. IEEE, 2016, pp. 130–139.

[39] A. Tomlinson, J. Bryans, and S. A. Shaikh, "Using a one-class compound classifier to detect in-vehicle network attacks," in *Proceedings of the genetic and evolutionary computation conference companion*, 2018, pp. 1926–1929.

[40] A. Tomlinson, J. Bryans, S. A. Shaikh, and H. K. Kalutarage, "Detection of automotive can cyber-attacks by identifying packet timing anomalies in time windows," in *2018 48th Annual IEEE/IFIP international conference on dependable systems and networks workshops (DSN-W)*. IEEE, 2018, pp. 231–238.

[41] A. Van Herrewege, D. Singelee, and I. Verbauwhede, "Canauth-a simple, backward compatible broadcast authentication protocol for can bus," in *ECRYPT workshop on Lightweight Cryptography*, vol. 2011. ECRYPT, 2011, p. 20.

[42] M. E. Verma, M. D. Iannacone, R. A. Bridges, S. C. Hollifield, P. Moriano, B. Kay, and F. L. Combs, "Addressing the lack of comparability & testing in can intrusion detection research: A comprehensive guide to can ids data & introduction of the road dataset," *arXiv preprint arXiv:2012.14600*, 2020.

[43] H. Wen, Q. A. Chen, and Z. Lin, "{Plug-N-Pwned}: Comprehensive vulnerability analysis of {OBD-II} dongles as a new {Over-the-Air} attack surface in automotive {IoT}," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 949–965.

[44] M. Wolf, *Computers as components: principles of embedded computing system design*. Elsevier, 2012, ch. 9.3 Networked control systems in cars and airplanes.

[45] S. Woo, H. J. Jo, and D. H. Lee, "A practical wireless attack on the connected car and security protocol for in-vehicle can," *IEEE Transactions on intelligent transportation systems*, vol. 16, no. 2, pp. 993–1006, 2014.

[46] H. Yang, M.-Z. Liu, Y.-H. Xu, Y.-J. Wu, and Y.-N. Xu, "Research of automotive ethernet security based on encryption and authentication method," *Int. J. Comput. Theory Eng*, vol. 11, no. 1, pp. 1–5, 2019.

[47] 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.