



Enhancing In-Vehicle Network Security Through Bitstream Feature Extraction-Based Intrusion Detection

Md Rezanur Islam

arupreza@sch.ac.kr

Department of Software Convergence,
Soonchunhyang University
Chungnam, Asan-si, South Korea

Muminov Ibrokhim Botir Ugli

theibrokhim@sch.ac.kr

Department of Computer Software
Engineering, Soonchunhyang
University
Chungnam, Asan-si, South Korea

Insu Oh

KANGBIN YIM

catalyst32@sch.ac.kr
yim@sch.ac.kr

Department of Information Security
Engineering, Soonchunhyang
University
Chungnam, Asan-si, South Korea

ABSTRACT

The in-vehicle network (IVN) is highly vulnerable due to its inherent structure, and the continuous introduction of new features in next-generation vehicles only exacerbates this issue. To address this problem, a proposed intrusion detection approach for in-vehicle networks is based on bitstream feature extraction and autocorrelation analysis. This approach aims to overcome the limitations associated with memory utilization and processing overhead vulnerability by comparing payload features using nibble. The Controller Area Network (CAN) data is highly susceptible, and organizations worldwide are working on developing Intrusion Detection Systems (IDSs) to protect vehicles. Nibbles are four-bit units of digital information, and autocorrelation measures the correlation of a variable with itself over time. In the proposed approach, the CAN payload is processed using nibble feature extraction and fed into a deep-learning CNN for attack classification. This approach presents a promising solution for enhancing the security of in-vehicle networks, by utilizing advanced machine-learning techniques to improve the security of vehicle systems.

CCS CONCEPTS

- In-vehicle Network(IVN); • Controller Area Network(CAN);
• Intrusion Detection System(IDS);

KEYWORDS

Feature Extraction, Nibble, Autocorrelation, CNN.

ACM Reference Format:

Md Rezanur Islam, Muminov Ibrokhim Botir Ugli, Insu Oh, and KANGBIN YIM. 2023. Enhancing In-Vehicle Network Security Through Bitstream Feature Extraction-Based Intrusion Detection. In *2023 Fifteenth International Conference on Contemporary Computing (IC3-2023) (IC3 2023)*, August 03–05, 2023, Noida, India. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3607947.3607989>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

IC3 2023, August 03–05, 2023, Noida, India

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0022-4/23/08...\$15.00

<https://doi.org/10.1145/3607947.3607989>

1 INTRODUCTION

With continuous advancements in information technology and automobile manufacturing, vehicles are becoming increasingly sophisticated. They come equipped with features that simplify drivers' lives and elevate their driving experience to the next level. Compared to traditional vehicles, which had no connectivity with the external environment, modern vehicles can easily communicate with the outside world through wireless connectivity such as Wi-Fi, Bluetooth, GPS, and navigation systems, advanced driver assistance systems (ADAS), and vehicle-to-everything (V2X) communication [5]. However, as vehicles become more advanced, they also become more vulnerable to cyber attacks, which raises concerns related to driver safety. It is evident that the Control Area Network (CAN), the most widely used in-vehicle bus communication protocol, is significantly susceptible to cyber attacks due to its lack of security features such as encryption and authentication [7]. The limited data frame of the CAN also makes it difficult to implement security features. Intrusion detection systems (IDSs) are effective in protecting in-vehicle networks against cyber attacks. However, these systems use a significant amount of memory and may add extra processing overhead, potentially slowing down the vehicle's systems and affecting performance. To address these issues, we propose a new intrusion detection approach for in-vehicle networks based on bitstream feature extraction. In this research, we conducted a brief study on the payload to differentiate between normal and attack situations. We note that another study on in-vehicle network attack detection on the payload has been previously conducted [10], which was more direct in its approach. In this study, we compared the payload feature by nibble and applied the autocorrelation mathematical method to identify patterns. Our proposed approach aims to overcome the limitations associated with the conventional IDSs, which use brute force memory and processing power to detect attacks. By utilizing bitstream feature extraction, our approach can provide a more efficient and effective solution for intrusion detection in in-vehicle networks. We also applied deep-learning CNN for attack classification to achieve higher accuracy in attack detection. This approach has the potential to improve the security of in-vehicle networks, reducing the risk of cyber-attacks and enhancing the safety of drivers and passengers.

2 CONTROLLER AREA NETWORK SPECIFICATION

In the field of automotive electronics and embedded systems engineering, the Controller Area Network (CAN) bus protocol is widely used to facilitate communication between different electronic control units (ECUs) in a vehicle. In a CAN message, the payload typically consists of an 8-byte hexadecimal number. CAN payload can be segmented into four categories [13]: constant values, multi-values, counters, and check codes. Constant values represent static information or configuration settings that remain unchanged over time. Multi-values consist of a few bytes that are responsible for carrying the payload of the command and can contain two or more changing values within these types of signals. Counters are indicators that behave as cyclic counters inside a selected range and can serve as additional syntax checks or be intended to order longer signal records at the destination ECU(s) [15][6]. Check codes are typically the last signal within the payload and provide additional error detection and correction capabilities to ensure the message data is transmitted accurately and reliably. The purpose of these segments is to provide additional information about the message data and improve the efficiency and reliability of the communication on the CAN bus network. Understanding the workings of these segments is crucial to ensure effective and efficient communication between the different ECUs in a vehicle.

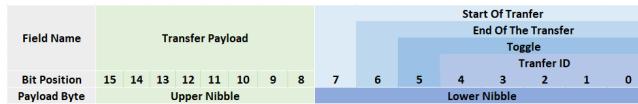


Figure 1: Controller Area Network Payload Specification [18]

The CAN payload structure and its various fields are well-documented in the literature [1]. The CAN (Controller Area Network) protocol is a widely-used communication standard in the automotive industry and other fields. In this article, we focus on the payload. In a CAN message, the payload contains the actual data being transmitted between devices. The CAN payload consists of 8 bytes of data, with each byte (or "octet") containing 8 bits of information, which can be either data or control information. Generally, CAN messages differ from manufacturer to manufacturer according to the CAN DBC file which is confidential, but here, we include an ideal payload configuration (used in UAVCAN) [18]. The control information includes fields such as the transfer ID, the toggle bit, and the start/end of transfer fields. The transfer ID is a 5-bit value that identifies the particular transfer within a CAN message. This allows multiple transfers to occur simultaneously over the same CAN bus without interfering with each other. The toggle bit is a control bit that alternates between 0 and 1 for each frame in a multi-frame transfer. This ensures the recipient device can detect if any frames have been lost or corrupted during transmission. The start and end of transfer fields indicate the beginning and end of a transfer. For single-frame transfers, these fields are always set to 1. For multi-frame transfers, the start of the transfer field is set to 1 for the first frame, while the end of the transfer field is set to 1 for the last frame.

3 RELATED WORKS

The more sophisticated the technology installed in vehicles become, the more ways adversaries can attack it, both remotely and physically. As mentioned earlier, the Controller Area Network (CAN) data is the most vulnerable part of a vehicle, as it lacks security features that can prevent a cyber attack, making it an easy target for opponents. To protect vehicles from unwanted intrusions, many organizations worldwide are actively conducting research on developing Intrusion Detection Systems (IDSs). For example, Linxi et al. [22] proposed a binarized neural network for in-vehicle network intrusion detection. The BNN uses binary values for inputs, weights, and activations, resulting in accelerated detection with low memory and energy usage. FPGAs were used to further accelerate the IDS, resulting in detection that was 3x faster than traditional IDS and 128x faster with FPGAs. Markus et al. [9] proposed a novel unsupervised learning approach, CANet, for intrusion detection on the CAN that outperformed previous machine learning methods. The CANet protocol is designed to operate on the CAN data signal space and displayed outstanding results in detecting specific signal manipulations where they introduce different types of attack data. Vita et al. [4] suggested a distance-based IDS utilizing an unsupervised Kohonen SOM network. This system has gained universal recognition in the security industry due to its high detection rate and low training time. Despite the sophisticated forms of the CAN dataset, their results have demonstrated efficacy and accuracy in detection, particularly in DoS and Spoofing attacks. In a study [19], hierarchical temporal memory (HTM) was suggested to create a distributed anomaly detection system for the in-vehicle network. This system predicts the next bit using the data stream of each CAN ID and calculates the anomalous score based on the incoming data and prediction to identify any potential attack. Article [2] proposed a method that combined federated random forest and blockchain to address poisoning attacks. However, their focus was on creating secure blockchain storage for the global model and did not address the issue of data heterogeneity. Furthermore, their approach was at the vehicle level in federated learning, whereas our proposed method is at the car manufacturer level where data labels are readily available.

In recent years, there has been significant attention given to in-vehicle network security, and a large number of research studies have been conducted. Our contribution is that, until now, no research has been conducted on the payload binary bit sequence, which could be used as an element for feature extraction in intrusion detection systems.

4 TYPES OF ADVERSARIAL ATTACKS IN CAN, DATA DESCRIPTION AND NEURAL NETWORK BACKGROUND

4.1 Adversarial Attacks

The Controller Area Network (CAN) bus protocol is widely used in the automotive industry for communication between electronic control units (ECUs) in a vehicle. However, the increasing use of this protocol has led to an increase in adversarial attacks that target the communication network. These attacks can be classified into three

main types: Fuzzing Attack Class, Dos Attack Class, and Replay Attack Class.

The Fuzzing Attack Class involves an attacker using a spoofed ID to compose a message with fabricated information. The message contains randomly generated package IDs, resulting in an increase in arbitrary ID amounts compared to other attack scenarios. This can cause abnormal anomalies in all nodes on the network [12].

The Dos Attack Class involves an attacker injecting high-priority packets into the bus at close time intervals, keeping the bus occupied and preventing lower-priority nodes from accessing the network properly. This can result in a denial of service (DoS) attack, causing delays and preventing legitimate packets from being transmitted [16]. These types of DoS attacks can cause a vehicle to cease responding to driving commands.

The Replay Attack Class assailants can observe data arrangements during a specific time interval and duplicate and repeat either the entire or a portion of these authentic message sequences. Each payload contains a valid CAN control message, which can potentially cause harm or unexpected behavior in vehicles. Detecting a replay attack is particularly difficult. In this type of attack, the attacker first observes the network traffic and filters out data from one or more randomly selected target node IDs. The attacker then stores this data along with its exact packet entry time, which is later used to reproduce or replay the same data at a future timestamp by injecting these packets into the network [14].

It is essential to detect and prevent these types of attacks as they can cause severe harm to the vehicle and its occupants. Therefore, various techniques have been proposed in the literature to mitigate these attacks, such as intrusion detection systems and secure communication protocols. However, further research is needed to develop effective countermeasures against these adversarial attacks in the CAN bus.

4.2 Data Description and Setup

Many researchers use an interfering device connected to an OBDII connector to find available PID (Parameter ID). However, our method differs as we collect data from the internal gateway where all integrated ECUs are connected [11] and shown in Figure 2. We tap interfering devices specifically on CAN High and CAN Low in the gateway and collect raw data. The main difference between the OBDII port and our method is that all ECUs are not connected in the former, making it impossible to capture all types of data. Raw CAN traffic includes both diagnostic response messages and regular CAN traffic messages, which we analyze to extract candidates that have the same value as the diagnostic response.

The PID is the same as defined in the J1979 standard, and vehicle status data can be collected using standard PIDs, regardless of the vehicle model. We used the PEAKCAN system as an interfacing device and Python Jupyter and Keras with TensorFlow as the backend for our investigation. Our experiment was conducted with Intel(R) Core (TM) i9-10900K CPU @ 3.70 GHz and NVIDIA GeForce RTX 2080 super.

4.3 Nural Network Bckground

The Convolutional Neural Network (CNN) model was first introduced in the 1980s by Yann LeCun and his colleagues as a method

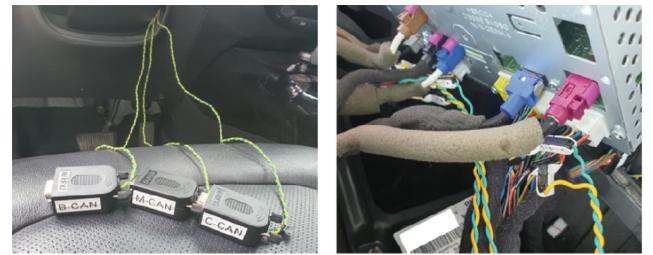


Figure 2: Data Collection Through CAN Gateway [11]

for image classification. Since then, it has evolved and become a powerful tool for processing various types of data [21]. One of the key features of CNN is its ability to automatically learn and extract features from the input data, without the need for manual feature engineering. CNNs have been successfully applied in various fields, including computer vision, speech recognition, natural language processing, and genomic analysis. Some notable applications include image recognition tasks such as object detection and classification, as well as speech recognition tasks such as voice identification. The Convolutional Neural Network (Conv2D) model is well suited for processing binary data consisting of 64 bits of 0 and 1. This model employs a series of filters to extract and learn key features from the input data. The filters are then convolved with the input, producing feature maps that are then subjected to additional filtering and pooling layers. The resulting output is a compressed representation of the original input data that can be used for classification or regression tasks. This approach has been successfully applied to a variety of binary data types, including images, audio, and genomic data.

5 FEATURE EXTRACTION METHOD

The term "nibble" refers to a unit of digital information that consists of four bits (or half a byte). A bit is the smallest unit of digital information and can represent either a 0 or a 1. A nibble is used to represent a single hexadecimal digit (0 to F). Nibbles are used in computer systems for a variety of purposes. For example, in some computer architectures, instructions are stored as nibbles. In addition, when data is transferred between computers, it is often sent in nibbles to minimize the amount of data that needs to be transferred. Nibbles can also be useful in data compression algorithms, as they allow for more efficient encoding of data. By compressing data into nibbles, it becomes possible to represent more data using fewer bits, which can lead to significant reductions in the amount of storage space needed to store data [3]. Overall, nibbles are a useful and commonly used unit of digital information that can help to optimize computer systems and reduce storage requirements. In Figure 3, we divided the 8-byte payload into nibbles for different scenarios, including attack-free, fuzzing, DoS, and replay. Fuzzing and DoS have their own distinct characteristics in specific time series sequences. The visualization for attack-free and replay scenarios is quite similar, but the density of bit changes is higher in the replay. Figure 3 represents only the changed bytes, which are the result of performing an XOR operation on the last byte and the previous byte.

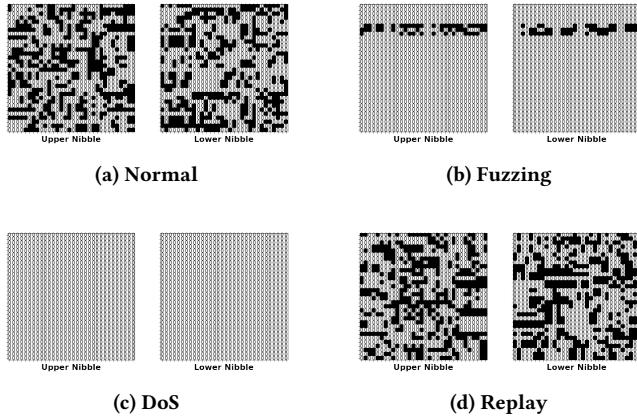


Figure 3: Controller Area Network Payload Nibble XOR Operation Visualization

After that Autocorrelation method is applied for finding the sequence of changing byte patterns. Autocorrelation is a statistical method used to determine the degree to which a variable is correlated with itself over time. It is a measure of how a variable's past values are related to its current or future values [20]. In other words, it is the correlation of a signal with a delayed copy of itself. Autocorrelation is used in a wide range of fields, including signal processing, finance, physics, and biology, among others. It is particularly useful in time series analysis and CAN is a time series data, where it is used to detect patterns and trends in data[17]. There are different types of autocorrelation, including lag-1 autocorrelation, which measures the correlation between a variable and its lagged value, and partial autocorrelation, which measures the correlation between a variable and its lagged value, controlling for the effect of intermediate values. Autocorrelation can also be used to test for randomness in a data set. If the autocorrelation coefficients are close to zero, it suggests that the data is random, while if there is a significant autocorrelation, it suggests that there is some underlying pattern or trend in the data. The mathematical definition shown below from [8] :

$$c_k = \sum_n a_{n+k} \cdot \bar{v}_n \quad (1)$$

The `numpy.correlate()` function calculates the cross-correlation of two 1-dimensional sequences, a and v . This is achieved by computing the sum of products between the elements of sequence a and the complex conjugate of sequence v , with the sequences being zero-padded where necessary. The function offers three modes for calculating the correlation ('valid', 'same', and 'full'), with 'valid' being the default mode but we used it here full. The function returns an array representing the discrete cross-correlation of a and v .

In Figure 4, the plot represents the autocorrelation values on the y-axis and the lag values on the x-axis. The lag measures the time delay between the original signal and the autocorrelated signal. The normal situations nibbles are compared with fuzzing, DoS, and replay attacks. The range of lag is similar for all scenarios because the same amount of data was used. However, the autocorrelation coefficient of each attack follows its unique pattern and contrasts

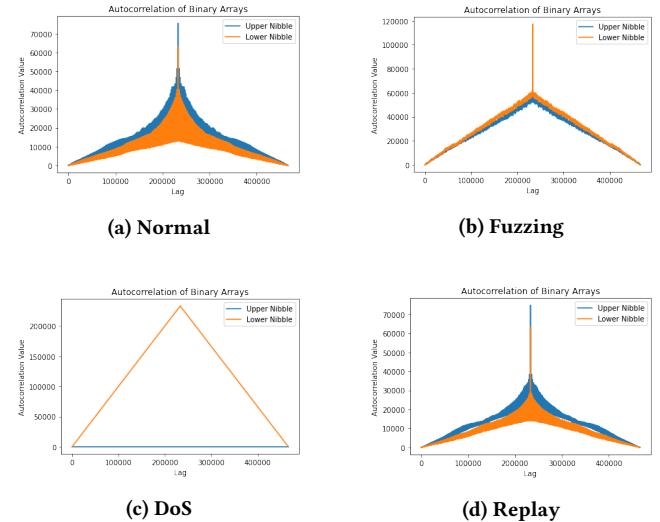


Figure 4: Autocorrelation coefficient for Attack Free, Fuzzing, DoS, and Replay

with the replay attack. As mentioned in the previous section, a replay attack was generated through the vehicle's own data. Therefore, the structure of the graph between normal and replay attacks is the same, but the autocorrelation coefficient has less variance than normal due to data repetition. Overall, the autocorrelation analysis provides insights into the properties of the signal and helps in identifying different types of attacks.

6 RESULT AND DISCUSSION

The paper focuses on detecting different types of adversarial attacks in the Controller Area Network (CAN) Bus. The proposed model is a Convolutional Neural Network (CNN) that takes input data in the form of two-dimensional arrays of shape (10, 2, 1), where 10 represents the time window, and 2 represents the number of signals. The model consists of three sets of Conv2D layers with increasing numbers of filters, followed by MaxPooling2D and Dropout layers to reduce overfitting. The output from the Conv2D layers is flattened and passed through two Dense layers with ReLU activation functions and a final Softmax output layer with four classes. The model achieved high accuracy and performance in detecting the four types of attacks: Normal, DoS, Fuzz, and Replay, with an accuracy of 0.98 and a ROC AUC score of 0.984 Figure 5. The precision, recall, and F1 score of each class were also reported in Table 1, showing high values for all classes. The high performance of the model suggests that it can effectively detect and classify different types of adversarial attacks on the CAN Bus.

The proposed model can be useful in the field of cybersecurity, especially in the automotive industry, where the CAN Bus is widely used. The detection of adversarial attacks on the CAN Bus can prevent malicious actors from accessing sensitive systems and causing harm. The model can be integrated into existing security measures to enhance the security of vehicles and prevent potential cyber-attacks. The proposed CNN model is effective in detecting different types of adversarial attacks on the CAN Bus, with high accuracy

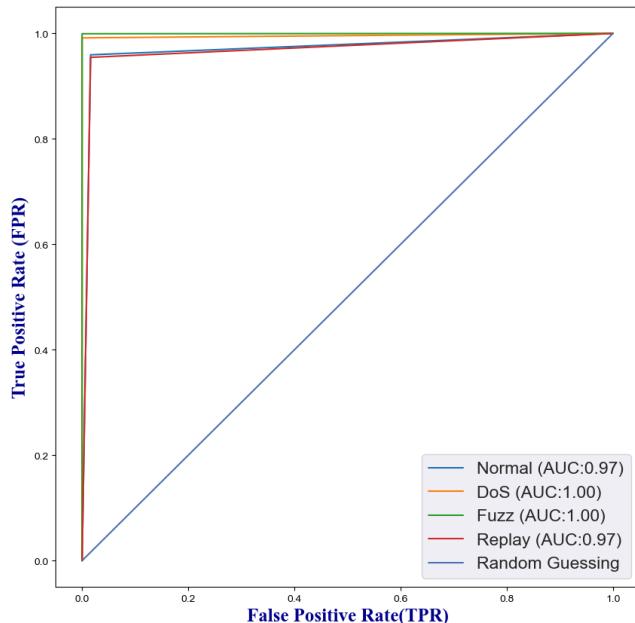


Figure 5: ROC Curve for Evaluating Performance

Table 1: CNN Model Performance Evaluation Table

Category	Precision	Recall	F1-score	Data Amount
Normal	0.95	0.96	0.95	11853
Fuzzing	1.00	0.99	1.00	11530
DoS	1.00	1.00	1.00	11605
Replay	0.95	0.95	0.95	11642
Accuracy		0.98		46630
AUC ROC		0.98		

and performance. The model can be useful in enhancing the cybersecurity of vehicles and preventing potential malicious attacks. Further research can be done to optimize the model and improve its performance in real-world scenarios.

7 CONCLUSION

The Controller Area Network (CAN) protocol is a widely used communication standard in various industries, especially the automotive sector. Although CAN provides a flexible and efficient means of transmitting data among devices, its vulnerability is a significant concern that threatens its reliability and security. Our research aimed to find a reliable way of securing the CAN protocol, and we developed an approach that detected malicious messages with an impressive success rate of 98%, outperforming other methods. However, it's worth noting that our approach only focused on payload analysis. In the future, we plan to expand our intrusion detection system to include arbitrary CAN ID correlations, in addition to CAN ID correlated payloads. This will help us achieve a more comprehensive and accurate detection of malicious messages

on the CAN network. Ultimately, our research highlights the importance of addressing the security concerns associated with the CAN protocol to ensure the reliability and safety of communication among devices.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea (NRF), which is funded by the Korean government (MSIT) under Grant Number 2021R1A4A2001810, and by Institute for Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2022-0-01197, Convergence security core talent training business (SoonChunHyang University).

REFERENCES

- [1] 1991. BOSCH CAN Specification Version 2.0. (1991). <http://esd.cs.ucr.edu/webscan20.pdf#=>
- [2] Ibrahim Aliyu, Marco Carlo Feliciano, Selinde Van Engelenburg, Dong Ok Kim, and Chang Gyoong Lim. 2021. A Blockchain-Based Federated Forest for SDN-Enabled In-Vehicle Network Intrusion Detection System. *IEEE Access* 9 (2021), 102593–102608. <https://doi.org/10.1109/ACCESS.2021.3094365>
- [3] Guilherme Arrozo, José Monteiro, and Arlindo Oliveira. 2018. Digital Representation of Information. In *Computer Architecture*. WORLD SCIENTIFIC, 1–30. https://doi.org/10.1142/9789813238343_0001
- [4] Vito Santa Barletta, Danilo Caivano, Antonella Nannavecchia, and Michele Scalera. 2020. Intrusion Detection for in-Vehicle Communication Networks: An Unsupervised Kohonen SOM Approach. *Future Internet* 12, 7 (2020). <https://doi.org/10.3390/fi12070119>
- [5] Carolina Blanch, Sofie Pollin, Gauthier Lafruit, Antoine Dejonghe, and Gregory Lenoir. 2007. Channel-Aware Rate Adaptation for Energy Optimization and Congestion Avoidance. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*. IEEE, I–821–I–824. <https://doi.org/10.1109/ICASSP.2007.366034>
- [6] Alessio Buscemi, Ion Turcanu, German Castignani, Romain Crunelle, and Thomas Engel. 2021. CANMatch: A Fully Automated Tool for CAN Bus Reverse Engineering Based on Frame Matching. *IEEE Transactions on Vehicular Technology* (2021). <https://doi.org/10.1109/TVT.2021.3124550>
- [7] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. 2011. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *Proceedings of the 20th USENIX Conference on Security* (San Francisco, CA) (SEC’11). USENIX Association, USA, 6.
- [8] NumPy Community. 2023. `numpy.correlate` – NumPy v1.24 Manual. <https://numpy.org/doc/stable/reference/generated/numpy.correlate.html>. (2023). Accessed: April 18, 2023.
- [9] Markus Hanselmann, Thilo Strauss, Katharina Dormann, and Holger Ulmer. 2020. CANet: An Unsupervised Intrusion Detection System for High Dimensional CAN Bus Data. *IEEE Access* 8 (jun 2020), 58194–58205. <https://doi.org/10.1109/ACCESS.2020.2982544> arXiv:1906.02492
- [10] Md Rezanur Islam, Insu Oh, Munkhdelgerekh Batzorig, Myoungsu Kim, and Kangbin Yim. 2022. Wavelet Transform Based PID Sequence Analysis for IDS on CAN Protocol. Vol. 2. 85–96. https://doi.org/10.1007/978-3-031-08819-3_9
- [11] Yeji Koh, Seoyeon Kim, Yoonji Kim, Insu Oh, and Kangbin Yim. 2022. Efficient CAN Dataset Collection Method for Accurate Security Threat Analysis on Vehicle Internal Network. *Lecture Notes in Networks and Systems* 496 LNNS (2022), 97–107. https://doi.org/10.1007/978-3-031-08819-3_10_COVER
- [12] Hyeryun Lee, Kyunghee Choi, Kihyun Chung, Jaemin Kim, and Kangbin Yim. 2015. Fuzzing CAN Packets into Automobiles. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*. IEEE, 817–821. <https://doi.org/10.1109/AINA.2015.274>
- [13] 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 *2017 15th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 57–5709. <https://doi.org/10.1109/PST.2017.00017>
- [14] Chung-Wei Lin and Alberto Sangiovanni-Vincentelli. 2012. Cyber-Security for the Controller Area Network (CAN) Communication Protocol. In *2012 International Conference on Cyber Security*. IEEE, 1–7. <https://doi.org/10.1109/CyberSecurity.2012.7>
- [15] Mirco Marchetti and Dario Stabili. 2019. READ: Reverse Engineering of Automotive Data Frames. *IEEE Transactions on Information Forensics and Security* 14, 4 (apr 2019), 1083–1097. <https://doi.org/10.1109/TIFS.2018.2870826>

- [16] Pal-Stefan Murvay and Bogdan Groza. 2017. DoS Attacks on Controller Area Networks by Fault Injections from the Software Layer. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*. ACM, New York, NY, USA, 1–10. <https://doi.org/10.1145/3098954.3103174>
- [17] C.J. Nunn and G.E. Coxson. 2008. Best-known autocorrelation peak sidelobe levels for binary codes of length 71 to 105. *IEEE Trans. Aerospace Electron. Systems* 44, 1 (jan 2008), 392–395. <https://doi.org/10.1109/TAES.2008.4517015>
- [18] UAVCAN Consortium. 2018. CAN bus transport layer – UAVCAN. https://legacy.uavcan.org/Specification/4_CAN_bus_transport_layer/. Accessed: April 18, 2023.
- [19] Chundong Wang, Zhentang Zhao, Liangyi Gong, Likun Zhu, Zheli Liu, and Xiaochun Cheng. 2018. A Distributed Anomaly Detection System for In-Vehicle Network Using HTM. *IEEE Access* 6 (1 2018), 9091–9098. <https://doi.org/10.1109/ACCESS.2018.2799210>
- [20] Xinmin Deng and Pingzhi Fan. 1999. New binary sequences with good aperiodic autocorrelations obtained by evolutionary algorithm. *IEEE Communications Letters* 3, 10 (oct 1999), 288–290. <https://doi.org/10.1109/4234.798020>
- [21] Akos Zarandy, Csaba Rekeczky, Peter Szolgay, and Leon O. Chua. 2015. Overview of CNN research: 25 years history and the current trends. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 401–404. <https://doi.org/10.1109/ISCAS.2015.7168655>
- [22] Linxi Zhang, Xuke Yan, and Di Ma. 2022. A Binarized Neural Network Approach to Accelerate in-Vehicle Network Intrusion Detection. *IEEE Access* 10, November (2022), 123505–123520. <https://doi.org/10.1109/ACCESS.2022.3208091>

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009