POLITECNICO DI TORINO
Repository ISTITUZIONALE

CANDoSA: A Hardware Performance Counter-Based Intrusion Detection System for DoS Attacks on Automotive CAN Bus

Original
CANDoSA: A Hardware Performance Counter-Based Intrusion Detection System for DoS Attacks on Automotive CAN Bus / Oberti, F., Di Carlo, S., Savino, A.. - ELETTRONICO. - (2025), pp. 1-5. (31st IEEE International Symposium on On-Line Testing and Robust System Design, IOLTS 2025 Ischia (ITA) 07-09 July 2025) [10.1109/iolts65288.2025.11116886].

Availability:
This version is available at: 11583/3006131 since: 2025-12-23T11:03:08Z

Publisher:
Institute of Electrical and Electronics Engineers

Published
DOI:10.1109/iolts65288.2025.11116886

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2025 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

04 November 2026

CANDoSA: A Hardware Performance Counter-Based Intrusion Detection System for DoS Attacks on Automotive CAN bus

Franco Oberti
Dumarey Softronix
Dumarey
Torino, Italy
ORCID: 0000-0001-7974-9505

Stefano Di Carlo
Control and Computer Eng. Dep.
Politecnico di Torino
Torino, Italy
ORCID: 0000-0002-7512-5356

Alessandro Savino
Control and Computer Eng. Dep.
Politecnico di Torino
Torino, Italy
ORCID: 0000-0003-0529-7950

***Abstract*—The Controller Area Network (CAN) protocol, essential for automotive embedded systems, lacks inherent security features, making it vulnerable to cyber threats, especially with the rise of autonomous vehicles. Traditional security measures offer limited protection, such as payload encryption and message authentication. This paper presents a novel Intrusion Detection System (IDS) designed for the CAN environment, utilizing Hardware Performance Counters (HPCs) to detect anomalies indicative of cyber attacks. A RISC-V-based CAN receiver is simulated using the gem5 simulator, processing CAN frame payloads with AES-128 encryption as FreeRTOS tasks, which trigger distinct HPC responses. Key HPC features are optimized through data extraction and correlation analysis to enhance classification efficiency. Results indicate that this approach could significantly improve CAN security and address emerging challenges in automotive cybersecurity.**

***Index Terms*—Security, CAN Networks, Intrusion Detection Systems, Hardware Performance Counters, Automotive**

I. Introduction

Modern vehicles are now interconnected platforms capable of semi-autonomous decision-making and Over-the-Air (OTA) updates. This integration of automotive engineering and information technology necessitates a rethinking of vehicle architectures to ensure functionality, performance, efficiency, and resilience.

Vehicle communication utilizes networks such as Controller Area Network (CAN), Long-Term Evolution (LTE), Fifth Generation Mobile Network (5G), and Ethernet to enable advanced features like Advanced Driver Assistance Systems (ADAS), Vehicle-to-Everything (V2X), and cloud-based services [1], [2]. However, this connectivity increases vulnerability to cyber threats [3], [4], including vehicle hijacking, Electronic Control Units (ECUs) manipulation, and ransomware attacks [5]. The shift to Software Defined Vehicle (SDV) introduces additional risks such as Denial-of-Service (DoS) attacks and data injection, making robust, multi-layered cybersecurity frameworks essential [6].

Regulations like United Nations Regulation No. 155 (UNR 155) and United Nations Regulation No. 156 (UNR 156) require manufacturers to implement Cybersecurity Management Systems (CSMSs) to ensure compliance [7]. Standards such as ISO/SAE 21434 Road Vehicles – Cybersecurity Engineering (ISO 21434) emphasize the need for intrusion detection to protect communication networks [8].

A common cybersecurity approach includes deploying Intrusion Detection Systems (IDSs) that analyze network traffic in real-time to identify threats. These systems leverage signature-based detection, anomaly detection, and machine learning to monitor networks like CAN and Ethernet [9], while host-based IDSs enhance security by tracking critical ECUs modifications.

This paper presents research on a novel IDS aimed at improving attack detection on CAN networks by assessing Hardware Performance Counters (HPCs) deviations in application execution. We use a RISC-V microprocessor as a representative architecture for next-generation automotive systems [10].

The paper is structured as follows: Section II provides background information, Section III describes the IDS framework, Section IV outlines the simulation environment, Section V discusses results, and Section VI concludes the paper.

II. Background

Automotive IDSs encompass signature-based, anomaly-based, and hybrid approaches for vehicle cybersecurity [11]. Signature-based systems detect known threats using predefined patterns, while anomaly-based solutions leverage Machine Learning (ML) and Artificial Intelligence (AI) to identify novel attacks through behavioral analysis [12]. Hybrid approaches combine these methods for comprehensive protection, though they face challenges in balancing detection accuracy with resource constraints.

This work was supported by Project SERICS through the MUR National Recovery and Resilience Plan, funded by the European Union NextGenerationEU under Grant PE00000014, project COLTRANE-V funded by the Ministero dell'Università e della Ricerca within the PRIN 2022 program (D.D.104 - 02/02/2022), and Project Vitamin-V funded by the European Union: Project 101093062.

Recent advances have integrated sophisticated ML techniques, particularly for CAN bus monitoring. The CAN-BERT model [13] applies language modeling to analyze network traffic patterns, while traditional classifiers like Support Vector Machines (SVMs) and Deep Neural Networks (DNNs) detect various attacks, including DoS and impersonation attempts [14], [15].

Vehicle security employs both network-based IDSs and Host-Based Intrusion Detection Systems (HIDSs). Network solutions monitor communication channels, while HIDSs protect individual ECUs through power signature analysis and host hardening techniques [16]. Commercial implementations combine these approaches with cloud computing and big data analytics for real-time threat detection [17], though high computational requirements can impact response times [11].

Key challenges include achieving high True Positive Rate (TPR) while maintaining low latency in resource-constrained environments. Future directions focus on integrating hardware security modules (Trusted Platform Modules (TPMs), Hardware Security Modules (HSMs)) with software-based detection [17], implementing federated learning for privacy-preserved model training [14], and developing standardized evaluation frameworks [18]. Recent research also explores HPCs event counting for detecting malware and microarchitectural attacks [19], [20].

III. Hardware-Based Intrusion Detection Framework

This section outlines the framework of the IDS, aimed at developing a HIDS to identify attacks by detecting behavioral deviations in the application on the ECU and analyzing CAN data (see Figure 1). The approach consists of three phases that transform raw CAN data into an effective online inference model using HPC event counting:

1) **Data Collection**: Data is gathered from the CAN network and organized into samples, which may be benign or malicious. This setup facilitates the detection of deviations in the ECU 's behavior, signaling potential attacks. While it limits detectable scenarios, it effectively identifies intrusion events like DoS attacks, which overwhelm ECU functionality, and frame spoofing attacks, where false messages are injected.
2) **Frame Processing and HPC Logging**: The CAN data is processed by an application on the ECU, activating the Perfomance Monitoring Unit (PMU) to log HPC values. This logging captures events during frame processing, providing insights into the ECU 's state and performance. Currently, HPC data is collected only at execution's end, limiting real-time detection but remaining crucial for identifying performance deviations [21].
3) **Offline Training and Online Inference**: The classification model undergoes offline training, processing HPC values through feature selection to identify key features and optimize classifier parameters. Once trained, the inference model continuously monitors CAN data to detect attacks based on deviations in the ECU 's behavior.

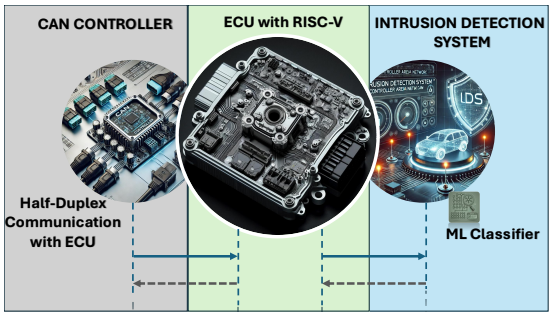


Fig. 1: General workflow for attack detection.

A. Classification Framework

The classification step is crucial for ensuring a fair analysis of HPC data. Initially, the raw data undergoes preprocessing to standardize and normalize features, eliminating discrepancies from differing scales and filtering out irrelevant data. This process enhances comparability and lays a foundation for accurate analysis. Two key transformations are applied:

1) **Mean and Scale Transformation:** This reduces dispersion in HPC values, mitigating the impact of outliers from simulation timing or workload variations.
2) **Correlation Analysis for Feature Reduction:** This analysis identifies and removes highly correlated features, reducing dimensionality while retaining the most predictive variables. The refined dataset focuses on HPC features that distinguish normal from anomalous traffic.

The classification employs Binary Classification Models to differentiate regular traffic from attacks. We utilize an unsupervised One-Class Classifier trained solely on standard traffic data, which requires less training data and effectively detects anomalies by identifying deviations from the established dataset. This approach ensures robust detection without constant updates, enhancing our cybersecurity efforts.

IV. Simulation of ECU-CAN systems

To deploy a realistic ECU configuration, a CAN receiver integrated with a RISC-V architecture is implemented on the gem5 simulator. The setup is configured using `riscv/fs_linux.py`, which establishes a RISC-V Timing Simple CPU, DDR4 RAM, L1/L2 caches, and a 1.0 GHz clocked system, reflecting a typical embedded system. This controlled environment overcomes physical board limitations, enabling precise observation and manipulation of CAN communication and HPC triggering. The simulation runs in gem5's Full System (FS) mode, emulating critical hardware components and supporting complex software like Linux or Real-Time Operating System (RTOS) (e.g., FreeRTOS [22]).

CAN communication involves two components:

- **CAN Controller Transmitter:** A Python script reads datasets, constructs CAN frames, and writes them to a shared file. Frames are generated by parsing components

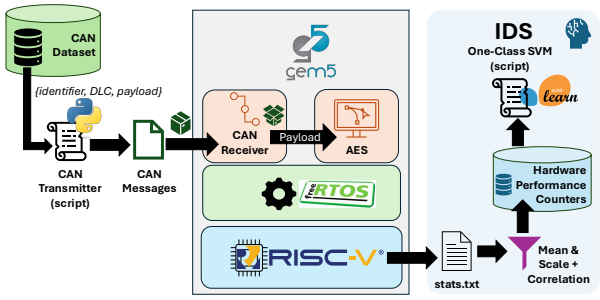


Fig. 2: gem5 [24] experimental environment

like identifiers, Data Length Code (DLC), and payloads and converting them into frames.

- **CAN Controller Receiver:** A *FreeRTOS* task reads and processes frame data from shared files, simulating memory-mapped access in the absence of a complete hardware counterpart.

Since the gem5 RISC-V microprocessor lacks a PMU, HPCs-like data are derived from gem5 logs [23]. The simulator tracks around 1200 architectural events in `stats.txt`, offering a broader dataset than physical PMU counters. A Python script parses these logs to extract and reshape HPC-like data for training and testing detection models, following preprocessing steps outlined in Section III-A.

V. Experimental Results

The experimental setup, shown in Figure 2, features an `AES-128` encryption task as a reference application. This task processes 16-byte plaintext blocks with a 16-byte key to produce ciphertext. It runs as a *FreeRTOS* task, receiving payloads as CAN frames from the CAN receiver described in IV. This setup simulates a realistic automotive environment by combining *FreeRTOS* and HPC evolution beyond the `AES` algorithm.

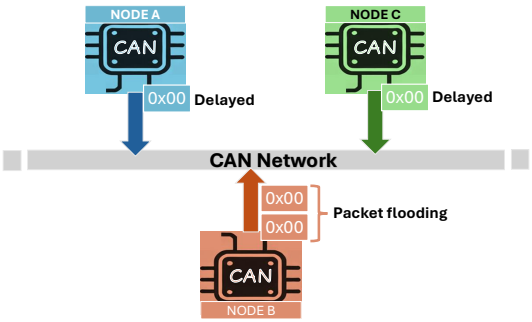


Fig. 3: Schematic of the DoS attack

The dataset, sourced from [29], was collected via the OBD-II port of a KIA SOUL car and includes regular (attack-free) data and three attack types: Fuzzy, Impersonation, and DoS. For this study, we focused on the DoS attack, where a malicious node injects CAN frames with an identifier of zero, causing arbitration delays for legitimate frames. The dataset contains 119,000 attack-free messages and 118,000 DoS attack messages.To evaluate the IDS framework, we used the unsupervised One-Class SVM [30], implemented via `sklearn` [31]. This model, trained on attack-free data, identifies anomalies by establishing a boundary around the normal class. Training utilized 20% to 95% of the attack-free data, with the remaining 5% reserved for testing alongside the full attack dataset.

After parameter tuning, the `OneClassSVM` was configured with an Radial Basis Function (RBF) kernel, $\nu = 0.2$, and $\gamma = auto$. This setup effectively detected anomalies within the dataset.

We implemented the two methods outlined in Section III-A to streamline our data. We standardized the input values, which ranged from 10^{-7} to 10^{10}, to achieve a mean of zero and a standard deviation of one, ensuring uniformity in our analysis. We computed correlation coefficients for each input parameter with the output parameter, excluding any that were uncalculable or did not meet the 0.9 threshold. This procedure ultimately informed the final selection in Table I. While the number of HPCs corresponds to the actual number of available counters, it is essential to note that the study's objective is to assess their effectiveness. It is plausible that by pinpointing a subset of HPCs, acceptable classification results can still be achieved with a limited number of physical counter registers [32].

Table I lists all selected logs, which include events that may not be available in standard RISC-V implementations, such as CVA6 [28], due to their optional nature. We also provide x86 similarity data [25]. Notably, the caches exhibit many similar events. Demand misses occur when the CPU requests an instruction not in the cache, requiring a fetch from lower memory levels. Read requests are a subset tracked when a read is issued to the gem5 cache component. This differentiation may not exist in real HPCs, but our preprocessing reveals that the information content does not fully overlap.

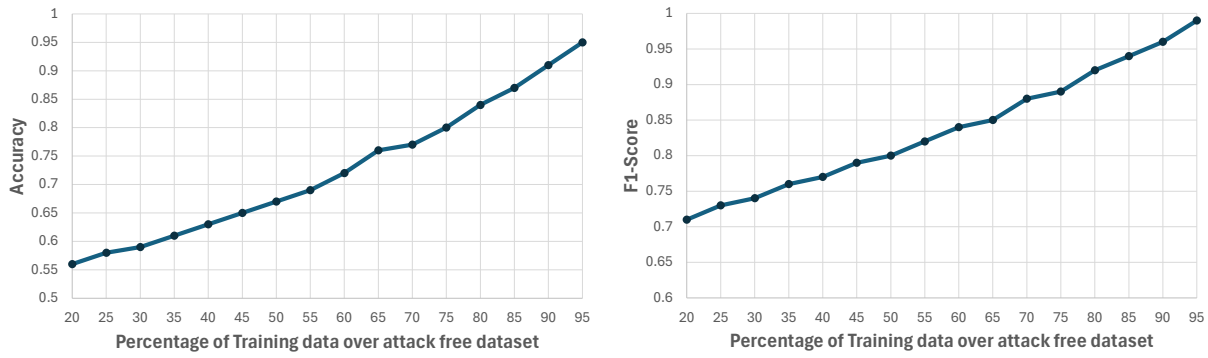
This study evaluates the effectiveness of one-class classification techniques in identifying various types of cyber attacks without prior knowledge of their characteristics. We focused on these methods' detection capabilities by varying the size of the attack-free training set, analyzing 1,500 CAN frames processed by the `AES` task. Smaller training set sizes, starting with 75 frames, yielded inconclusive results and have been excluded from the final manuscript for clarity and integrity.

In Figure 4, we present essential performance metrics, specifically accuracy and the F1 score (see equation 1), as we vary the percentage of the dataset utilized for training. The accuracy metric assesses the ratio of correctly predicted instances to the total number of predictions made. Meanwhile, the F1 score offers a balanced evaluation of precision and recall, making it particularly valuable in situations with imbalanced class distributions.

$$F1 = \frac{TP}{TP + \frac{1}{2}(FN + FP)} \tag{1}$$

TABLE I: Selected Events from gem5 logs (all start from `system.`)

gem5 Event	gem5 Meaning	RISC-V HPC Similarity	x86 HPC Similarity [25]
cpu.commitStats0.numInsts	Committed instructions	`minstret` (Retired instruction counter) [26]	INST_RETIRED.ANY
cpu.fetchStats0.numBranches	Fetched branch instructions	Branch instructions event (PULP) [27]	BR_INST_RETIRED.ALL_BRANCHES
cpu.dcache.demandHits::cpu.data	Demand hits in the data cache	L1 D-Cache hit event (PULP) [27]	MEM_LOAD_RETIRED.L1_HIT
cpu.dcache.demandMisses::cpu.data	Demand misses in the data cache	L1 D-Cache miss event (PULP) [27]	MEM_LOAD_RETIRED.L1_MISS
cpu.dcache.ReadReq.hits::cpu.data	Read request hits in the data cache	Load access event (CVA6) [28]	MEM_LOAD_RETIRED.L1_HIT (subset)
cpu.dcache.ReadReq.misses::cpu.data	Read request misses in the data cache	Load access event (CVA6) [28]	MEM_LOAD_RETIRED.L1_MISS (subset)
cpu.dcache.WriteReq.hits::cpu.data	Write request hits in the data cache	Store access event (CVA6) [28]	MEM_STORE_RETIRED.L1_HIT
cpu.dcache.WriteReq.misses::cpu.data	Write request misses in the data cache	Store access event (CVA6) [28]	MEM_STORE_RETIRED.L1_MISS
cpu.icache.demandHits::cpu.inst	Demand hits in the instruction cache	L1 I-Cache hit event (PULP) [27]	ICACHE.HIT
cpu.icache.demandMisses::cpu.inst	Demand misses in the instruction cache	L1 I-Cache miss event (PULP) [27]	ICACHE.MISS
cpu.icache.ReadReq.hits::cpu.inst	Read request hits in the instruction cache	Instruction fetch event (CVA6) [28]	ICACHE.HIT (subset)
cpu.icache.ReadReq.misses::cpu.inst	Read request misses in the instruction cache	Instruction fetch event (CVA6) [28]	ICACHE.MISS (subset)
l2.demandHits::cpu.data	Demand hits in the L2 cache	L2 cache hit event (if implemented in PULP/CVA6)	MEM_LOAD_RETIRED.L2_HIT
l2.demandMisses::cpu.inst	Demand misses in the L2 cache (instructions)	L2 cache miss event (if implemented in PULP/CVA6)	MEM_LOAD_RETIRED.L2_MISS (for instructions)
l2.demandMisses::cpu.data	Demand misses in the L2 cache (data)	L2 cache miss event (if implemented in PULP/CVA6)	MEM_LOAD_RETIRED.L2_MISS (for data)
l2.demandMisses::total	Total demand misses in the L2 cache	Total L2 cache miss event (if implemented in PULP/CVA6)	L2_RQSTS.MISS



(a) Attack detection Accuracy

(b) Attack detection F1-score

Fig. 4: IDS performances on DoS attack, considering a training set from 20% to 95% of the attack-free dataset in [29]

The results show that the one-class classifier effectively identifies all malicious samples, but its performance heavily relies on the training set size. Achieving 90% accuracy requires over 75% of the training data (between 75% and 80% in Fig. 4a). This limitation means that without a sufficiently large and diverse training set, the model may misclassify legitimate traffic as malicious, as indicated by the F1-score in Fig. 4b, which highlights the risk of high False Positive Rate (TPR) with inadequate training data.

Additionally, the classifier's ability to detect attacks depends on accumulating HPC data over time, preventing instantaneous or real-time intrusion detection. This latency poses challenges for immediate threat response and concerns environments requiring prompt action against security breaches, which will be addressed in future work.

While the current classifier does not support real-time IDS deployment, it demonstrates that malicious activities can cause significant deviations in application processing patterns. Understanding these deviations could enhance future detection algorithms and overall cybersecurity measures.

VI. Conclusion

This paper introduced a novel IDS approach utilizing HPCs for detection. Initial findings demonstrate the feasibility of detecting attack CAN data, although detection quality remains limited. The primary challenge is the need for a large dataset to train an effective model, indicating further experiments are necessary. Future work should explore various attacks, applications, and more complex RTOS scenarios.

Additionally, research will focus on adapting IDS capabilities for safety-critical real-time embedded systems, a vital step toward developing a comprehensive IDS. The ultimate aim is to create an advanced IDS that integrates CAN bus anomaly detection and HIDS into a unified module.

Acknowledments

This paper is based on the work conducted by Gaspard Henri Guy Michel as part of their Master's thesis at Politecnico di Torino. Their dedication and research have significantly contributed to the findings and insights.

REFERENCES

[1] A. Brown and B. Green, "The evolution of in-vehicle networks: From can to ethernet," *Journal of Automotive Networking*, vol. 15, no. 3, pp. 567–580, 2022.

[2] J. Miller and D. Smith, "A survey on vehicle communication protocols: Can, lin, and ethernet," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 5, pp. 1123–1135, 2023.

[3] M. Williams, "Cybersecurity threats in modern vehicles," *IEEE Security & Privacy*, vol. 21, no. 4, pp. 45–52, 2023.

[4] K. Anderson and L. Schmidt, "The security implications of remote firmware updates and diagnostics in modern vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 121–135, 2023.

[5] M. Davis and A. Lopez, "Case studies of vehicle hijacking and ecu manipulation attacks," *Journal of Automotive Cyber Threats*, vol. 9, no. 4, pp. 98–112, 2024.

[6] T. Nguyen and R. Patel, "Denial-of-service and spoofing attacks in automotive networks: A security framework," *Journal of Automotive Cybersecurity*, vol. 8, no. 1, pp. 25–40, 2024.

[7] E. Jones, "The role of intrusion detection systems in automotive cybersecurity," *Automotive Cybersecurity Review*, vol. 8, no. 2, pp. 30–45, 2024.

[8] L. Schmidt and K. Anderson, "Automotive cybersecurity engineering: Practical applications of iso 21434," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 2, pp. 234–249, 2024.

[9] S. Kim and W. Chang, "The role of intrusion detection systems in automotive cybersecurity," *IEEE Security & Privacy*, vol. 22, no. 3, pp. 50–65, 2023.

[10] L. Cuomo, C. Scordino, A. Ottaviano, N. Wistoff, R. Balas, L. Benini, E. Guidieri, and I. M. Savino, "Towards a risc-v open platform for next-generation automotive ecus," in *2023 12th Mediterranean Conference on Embedded Computing (MECO)*, 2023, pp. 1–8.

[11] J. Luo, Y. Jiang, J. Wu, Z. Ding, and X. Zheng, "A survey on intrusion detection for connected and autonomous vehicles," *IEEE Internet of Things Journal*, vol. 10, no. 13, pp. 11 841–11 869, 2023.

[12] J. D. Frechette, P. Tsai, and E. Lam, "Systems and methods for in-vehicle communication and control," US Patent US10 630 699B2, 2020. [Online]. Available: https://patents.google.com/patent/US10630699B2/en

[13] N. Alkhatib, M. Mushtaq, H. Ghauch, and J.-L. Danger, " CAN-BERT do it? Controller Area Network Intrusion Detection System based on BERT Language Model ," in *2022 IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA)*. Los Alamitos, CA, USA: IEEE Computer Society, Dec. 2022, pp. 1–8. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/AICCSA56895.2022.10017800

[14] B. S. Bari, K. Yelamarthi, and S. Ghafoor, "Intrusion detection in vehicle controller area network (can) bus using machine learning: A comparative performance study," *Sensors*, vol. 23, no. 7, p. 3610, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/7/3610

[15] B. Alzahrani, A. Alshahrani, A. Alzahrani, F. Alshahrani, M. Alshahrani, M. Alzahrani, and S. Alzahrani, "A comprehensive survey of cyberattacks on evs: Research domains, attacks, defensive mechanisms, and verification methods," *Transportation Engineering*, vol. 12, p. 100187, 2023.

[16] S. Khandelwal and S. Shanker, "Phidias: Power signature host-based intrusion detection in automotive systems," in *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*, 2024. [Online]. Available: https://dl.acm.org/doi/10.1145/3689939.3695780

[17] B. Alzahrani, A. Alshahrani, A. Alzahrani, F. Alshahrani, M. Alshahrani, M. Alzahrani, and S. Alzahrani, "A comprehensive review of ai based intrusion detection system," *ICT Express*, 2024.

[18] M. L. Bouchouia, H. Khemissa, E. Gherbi, M. Tami, D. Lopes, N. Alkhatib, and M. Ayrault, "Cybersecurity metrics for ai-based in-vehicle intrusion detection systems," in *2024 IEEE Vehicular Networking Conference (VNC)*, 2024, pp. 269–270.

[19] E. Magliano, A. Carpegna, A. Savino, and S. D. Carlo, "A micro architectural events aware real-time embedded system fault injector," in *2024 IEEE 25th Latin American Test Symposium (LATS)*, 2024, pp. 1–6.

[20] D. Kasap, A. Carpegna, A. Savino, and S. Di Carlo, "Micro-architectural features as soft-error markers in embedded safety-critical systems: preliminary study," in *2023 IEEE European Test Symposium (ETS)*, 2023, pp. 1–5.

[21] A. P. Kuruvila, X. Meng, S. Kundu, G. Pandey, and K. Basu, "Explainable machine learning for intrusion detection via hardware performance counters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4952–4964, 2022.

[22] FreeRTOS, "Freertos documentation," 2024, https://www.freertos.org/Documentation/RTOS_book.html. [Online]. Available: https://www.freertos.org/Documentation/RTOS_book.html

[23] S. Dutto, A. Savino, and S. Di Carlo, "Exploring deep learning for in-field fault detection in microprocessors," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021, pp. 1456–1459.

[24] J. Lowe-Power, "Gem5 documentation," Gem5, 2024, https://www.gem5.org/documentation/. [Online]. Available: https://www.gem5.org/documentation/

[25] Intel Corporation, *Intel® 64 and IA-32 Architectures Software Developer's Manual*, 2021, document Number: 325384-074US. [Online]. Available: https://software.intel.com/content/www/us/en/develop/articles/intel-sdm.html

[26] RISC-V Foundation, *The RISC-V Instruction Set Manual, Volume II: Privileged Architecture*, 2021. [Online]. Available: https://riscv.org/technical/specifications/

[27] PULP Platform, "Pulp platform documentation," 2021, accessed: 2023-03-13. [Online]. Available: https://pulp-platform.org/

[28] OpenHW Group, "Cva6 user manual," 2021, accessed: 2023-03-13. [Online]. Available: https://github.com/openhwgroup/cva6

[29] 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)*, 2017, pp. 57–5709.

[30] A. M. Mahfouz, A. Abuhussein, D. Venugopal, and S. Shiva, "Network intrusion detection model using one-class support vector machine," in *Advances in Machine Learning and Computational Intelligence*. Springer, 2020, pp. 77–87.

[31] S. learn, "Scikit learn documentation," 2024, https://scikit-learn.org/stable/. [Online]. Available: https://scikit-learn.org/stable/

[32] C. P. Chenet, A. Savino, and S. Di Carlo, "Zero-day hardware-supported malware detection of stack buffer overflow attacks: An application exploiting the cv32e40p risc-v core," in *2025 IEEE 26th Latin American Test Symposium (LATS)*, 2025, pp. 1–6.