Revealing Embedded System Behaviors: A Comparative Analysis of Power Consumption and Hardware Performance Counters

Mohammed Mezaouli$^{1[ID]}$, Yehya Nasser$^{1[ID]}$, Samir Saoudi$^{1[ID]}$, and Marc-Oliver Pahl$^{[ID]}$

IMT Atlantique, Lab-STICC, UMR CNRS 6285, Brest, France
`firstname.secondname@imt-atlantique.fr`

Abstract. As computing evolves, analyzing system behavior has become as critical as optimizing performance and energy efficiency. Traditional methods of behavioral analysis often rely on software-level profiling or predefined signatures, which limit their applicability in dynamic and resource-constrained environments. This study takes a novel approach by identifying and characterizing system workloads based on distinct behavioral patterns, such as memory usage, arithmetic operations, and control flow. By leveraging hardware performance counters (HPCs) and power consumption metrics, this work demonstrates how hardware-level insights can reveal unique operational signatures. These findings highlight the potential of HPCs and power consumption to enhance the understanding of system behaviors, offering practical solutions for behavior identification that could be instrumental in detecting vulnerabilities or anomalies. The results underscore both the opportunities and challenges of using these metrics for behavioral analysis in embedded systems.

Keywords: HPC · Power Consumption · Behavior identification.

1 Introduction

Embedded systems are fundamental to modern technology, driving innovations across industries such as automotive [16], healthcare, and industrial automation [11]. These systems are defined by their constrained resources, real-time processing needs, and specialized applications. Understanding and characterizing the behavior of embedded systems has become crucial for optimizing performance, improving reliability, and enhancing security [8].

Behavioral identification [5] in embedded systems involves analyzing how these systems execute operations like memory accesses, arithmetic computations, and control flow management [13]. Traditional methods of behavioral analysis often rely on software-level profiling or static analysis tools. However, these approaches can introduce significant overhead, lack the granularity needed for detailed insights, or fail to capture the dynamic interplay between hardware and software. Advanced malware evasion techniques employ sophisticated obfuscation methods to bypass detection, further underscores the limitations of these traditional methods [12]. This calls for efficient, hardware-assisted techniques capable of providing real-time insights into embedded system behaviors [6].

HPCs and power consumption present a promising solution for behavior identification. HPCs, embedded within modern processors, offer fine-grained insights into low-level events such as cache hits, branch predictions, and memory accesses, enabling detailed execution pattern analysis without imposing significant performance overhead. Power consumption, in contrast, provides a holistic view of system activity, reflecting the aggregate behavior of hardware components during runtime. By combining these two metrics, it becomes possible to derive a comprehensive understanding of embedded system behaviors, particularly in scenarios involving threats like firmware tampering and unsecured interfaces [15] [17].

In this work, we investigate the potential of (HPCs) and power consumption metrics for behavioral identification in embedded systems, and compare their effectiveness in terms of identification efficiency. We focus on operational behaviors such as memory management, arithmetic operations, and control logic, as these metrics enable the characterization of normal system behavior.

Unlike traditional profiling methods, our approach emphasizes non-intrusive, hardware-based techniques that are efficient and scalable for resource-constrained environments. The ability to dynamically monitor and interpret system behavior in real time has broad applications, including vulnerability analysis and malware or anomaly detection without the need for large sample datasets [9]. The main contributions of this study are: (1) Demonstrating the feasibility of using HPCs and power consumption, in conjunction with Artificial Intelligence (AI), as effective metrics for behavioral analysis; (2) providing a comparative evaluation of HPCs and power consumption for behavioral recognition in embedded systems.

2 Related Works

HPCs are often debated in terms of their effectiveness for malware detection, despite their advantage of lower overhead. One limitation is that HPCs cannot be fully relied upon alone for detecting malware, as they are influenced by both low-level micro-architecture and software-level activities [19].

Leng et al. [10] suggest that relying on only two HPCs may be sufficient to ensure system reliability, offering a simpler yet effective approach. Similarly, Patel et al. [13] utilized HPCs for malware detection by implementing several machine learning classifiers on FPGA platforms. Their methodology involved collecting 32 different HPC features and subsequently reducing the dataset to 8 key features. Notably, they found that using the OneR algorithm with only the branch feature achieved a classification accuracy of 82.5%. This demonstrates the effectiveness of HPC-based features for malware detection.

Basu et al. [3] investigated impact of HPC interval measurement on model learning for malware detection. They also used Control Flow Graphs (CFGs) to assess the probability of matches between two malware samples. Their findings indicated that the use of instruction and branch HPCs led to a low probability of exact matches between malware. They concluded that HPCs are not deterministic because of the presence of background processes , meaning that malware detectors must rely on approximate matching rather than precise identification.

However, Pham et al. [14] utilized electromagnetic field measurements to detect malware, achieving an accuracy of 99% even when obfuscation techniques were applied. This underscores the effectiveness of side-channel information in malware detection. Similarly, Azmoodeh et al. [2] employed power consumption data to identify patterns and trained a K-Nearest Neighbors (KNN) model, achieving a detection accuracy of 95.65% in effectively detecting ransomware.

Various malware detection techniques are reviewed by Aslan et al. [1], with behavior-based detection yielding promising results. Hernandez et al. [7] demonstrated that using power consumption alone for malware detection is effective, achieving an F-score of 0.97, compared to relying solely on network data, which achieved an F-score of 0.94. This is because power-related data are significant.

Bridges et al [4] employed both supervised and unsupervised anomaly detection methods to detect rootkit malware through power consumption, with the supervised approach showing the best results. Similarly, Yang et al. [18] used the Gaussian Mixture Model (GMM) on Android devices to detect floating point attacks based on power consumption, proving its ability to detect malware or apps such as music and browsers due to its low complexity and efficiency in approximating arbitrary distributions, achieving 79% for malware detection.

All prior work relies solely on Hardware Performance Counters (HPCs) or power metrics, often combined with additional methods to enhance detection techniques. These approaches predominantly focus on malware datasets for training, which limits their detection capabilities to known malware, making it challenging to identify zero-day threats.

Our work examines the effectiveness of Hardware Performance Counters (HPCs) and power consumption metrics in detecting workload behavior, offering a novel approach that integrates these non-invasive and readily accessible data sources to identify behavioral anomalies. Unlike prior work, we explore how code optimization levels impact detection accuracy, enhancing model robustness and enabling generalization to zero-day threats. By focusing on behavior-based detection rather than relying on large datasets, our methodology marks a significant step toward developing a comprehensive, explainable behavior-based detection framework for embedded systems. As a proof-of-concept, our proposed model consumes raw data from HPCs or power consumption metrics to analyze and detect workload behavior. In the future, we aim to refine this method further to detect anomalies and malicious behavior, with a particular focus on improving its capacity to identify zero-day threats.

3 Methodology

This study focuses on ARM M4 processors, commonly used in resource-constrained embedded systems with bare-metal setups to optimize efficiency and minimize overhead. Using a bare-metal configuration ensures the behavioral characteristics observed, such as power consumption and HPC metrics, are directly tied to the workload, free from variability introduced by operating system-level factors. This approach not only mirrors real-world deployment scenarios but also provides a precise analysis of workload behavior. Furthermore, the insights gained form a foundation for extending the methodology to more complex environments, such as those involving real-time operating systems.

To validate our hypothesis that power consumption and HPCs can serve as identifiers of activity within embedded processors, we implemented an experimental methodology comprising four key steps: (A) execution of benchmark workloads on embedded processors, (B) dynamic power measurement during program execution, (C) HPC instrumentation, and (D) construction of a workload identifier based on the captured power traces and HPC data.

3.1 Preparation and Execution of Benchmark Workloads

Our first step involved preparing benchmark workloads that cover typical computational tasks on embedded processors: memory assignment, logic operations, and arithmetic computations—compiled for execution on an ARM Cortex-M4 processor, STM32F303 included as a target board with the Chipwhisperer Lite. Measurements were conducted in a controlled environment equipped with power measurement tools to ensure accuracy and repeatability.

Each workload generates power consumption traces and HPC data, influenced by parameters like iteration count and optimization level; benchmarks were executed with three different optimization levels. As a preliminary benchmark we selected four workloads with diverse algorithms to target different computation tasks:

1. **Bubble sort**: relies on comparison and swap operations, iterating through pairs of adjacent elements and rearranging them based on their relative values, leading to repetitive control-dependent branching and memory access.
2. **Median Filter**: primarily uses sorting and selection operations within a local window of pixels or data points, requiring frequent neighborhood data access and sorting to compute the median, making it memory-intensive.
3. **Matrix Multiplication**: consists of multiplication and addition operations performed on matrix elements. This involves heavy arithmetic operations and regular memory access patterns, often optimized through block processing to improve data locality.
4. **Factorial**: involves multiplication operations either recursively or iteratively, with a straightforward arithmetic progression where each element is multiplied by its predecessor, requiring minimal memory but heavy arithmetic.

3.2 Power Measurement and Characterization

Power traces are captured during workload execution and aligned with the corresponding program execution phases, we correlated specific workload behaviors with their power consumption profiles, building comprehensive power characterizations for analysis. As illustrated in Fig. 1, we measured power by compiling workloads with the three optimization levels and embedding a trigger within the code to prompt the oscilloscope ChipWhisperer Lite to begin measurement upon execution. This approach ensured accurate capture of power consumption solely during workload execution. The oscilloscope was configured to sample power at each clock cycle, synchronized with transistor switching events. For workloads exceeding the oscilloscope's 24,000-point buffer capacity, we repeated measurements with adjusted parameters to create a sliding window, covering the entire execution.

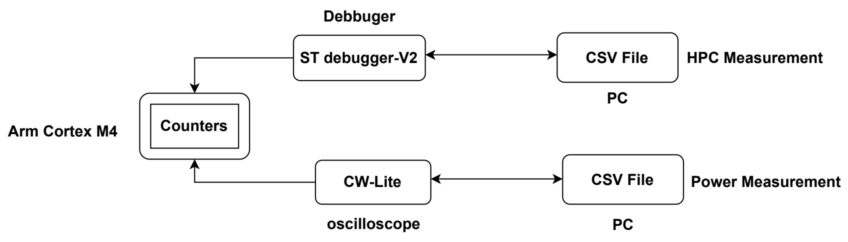

Fig. 1: Power Consumption and HPC Testbed

3.3 HPC Instrumentation and Characterization

We monitored HPCs during program execution to track key events like instruction execution, memory access, and cache utilization. Synchronizing HPC traces with workload execution phases allowed us to link workload behaviors to hardware activities. As illustrated in Fig. 1, we measured HPCs periodically using the ST-LINK/V2 debugger at a sampling frequency of 2 kHz. This method ensured no overhead during workload execution. We executed each workload with three optimization levels (O1, O2, O3), which affected instructions at various pipeline stages and execution time. These counters are available on Cortex-M3 and Cortex-M4 processors, but not on Cortex-M0. However, our work focuses exclusively on Cortex-M4.

1. **Cycle Counter (CYCCNT)**: Increments every CPU clock cycle.
2. **Load/Store Unit Cycle Counter (LSUCNT)**: Increments for each additional cycle taken by load and store instructions beyond their initial cycle.
3. **CPI Cycle Counter (CPICNT)**: Increments for each extra cycle required by multi-cycle instructions beyond the first cycle.
4. **Exception Overhead Cycle Counter (EXCCNT)**: Increments on each cycle spent handling exceptions (entry and exit).
5. **Folded-instruction Counter (FOLDCNT)**: Increments for each folded instruction, such as zero-cycle instructions like If-Then (IT) instructions and some NOPs (No Operation).
6. **Sleep Counter**: Increment on cycles associated with power saving mode.

3.4 Building the Workload Identifier Model

Finally, in the sake of comparison, we developed a workload and optimization identifier using two machine learning models, as illustrated in Fig. 2. A Multilayer Perceptron (MLP) which consists of three layers: an input layer of 25000 point matching the input feature dimension, two hidden layer with 64 neurons and 32 neurons respectively, and an output layer with 12 neurons (for the 12 class). ReLU activation functions are applied between layers, and the model is trained using the Adam optimizer with a learning rate of 0.001 and a 1D Convolutional Neural Network (CNN) that includes two convolutional layers with a kernel size of 5, stride of 1, and padding of 2, followed by ReLU activation and max-pooling. The convolutional output is flattened and fed into two fully connected layers: the first with 1000 neurons and the second with 12 neurons for classification. Both models are trained using the Adam optimizer, using minimize cross-entropy loss, with performance evaluated using validation accuracy. Models were trained on power consumption traces and HPC data, respectively, to map these metrics to computational activities within the embedded processor. This approach enables a non-invasive system capable of identifying workloads based solely on power consumption profiles or HPC measurements. The classification task assigns 12 class labels, representing four workloads across three optimization levels: Classes 0–2 correspond to Factorial at O1–O3, Classes 3–5 to Bubble Sort at O1–O3, Classes 6–8 to Matrix Multiplication at O1–O3, and Classes 9–11 to Median Filter at O1–O3. Finally, we evaluated workload identification without taking in consideration the opimization level (4 classes) to assess model performance on workload identifation.

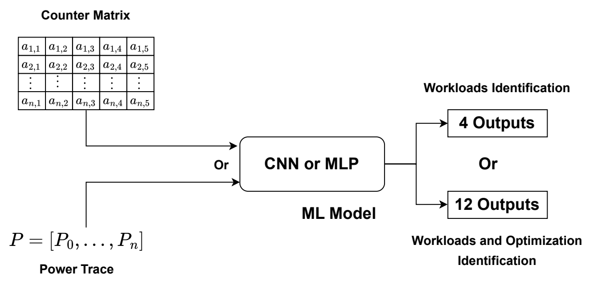
Counter Matrix
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,4}$ $a_{1,5}$
$a_{2,1}$ $a_{2,2}$ $a_{2,3}$ $a_{2,4}$ $a_{2,5}$
$a_{n,1}$ $a_{n,2}$ $a_{n,3}$ $a_{n,4}$ $a_{n,5}$
Or
CNN or MLP
ML Model
$P = [P_0, \ldots, P_n]$
Power Trace
Workloads Identification
4 Outputs
Or
12 Outputs
Workloads and Optimization Identification

Fig. 2: Our Proposed Model/Detector

4 Results and Discussions

In this section, we analyze the data, discuss the main results, and present the models proposed in our work. We examine HPCs and power consumption during workload execution.

Focusing on Fig.3, where LSU activity is shown, we note that workloads such as Factorial and Matrix Multiplication do not exhibit high activity during execution. In contrast, workloads like Bubble Sort and Median Filter present high activity in the number of loads and stores. In addition, we observed in Fig.4 that the folding instruction counters exhibited high activity during the execution of the Bubble Sort and Median Filter workloads, whereas they showed low or negligible activity for the Matrix Multiplication and Factorial workloads. This discrepancy arises from the fundamental differences in the algorithms control flow and the resulting interaction with the processor's instruction folding optimization. Bubble Sort and Median Filter are characterized by frequent conditional branches and data-dependent decision-making processes, which translate into numerous IT (If-Then) instructions and conditional branches. The ARM Cortex-M4 processor optimizes these instructions through instruction folding, effectively reducing execution cycles and improving pipeline efficiency. Consequently, the folding counters register high activity for these workloads. In contrast, Matrix Multiplication and Factorial are arithmetic-intensive algorithms with straightforward and predictable control flows involving simple loops and minimal branching. Their execution primarily consists of arithmetic operations with few, if any, foldable instructions. As a result, there are limited opportunities for the processor to apply instruction folding, leading to zero activity in the folding counters for these workloads

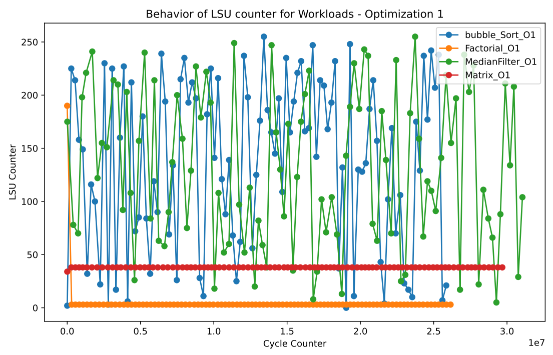

Fig. 3: LSU Counter Behavior

Motivated by the promising preliminary findings, we incorporated HPCs into our study to identify different workloads and their associated optimization levels on the ARM Cortex-M4 processor. Table 1 presents our experimental results. Initially, we utilized four HPC counters as inputs to a Multilayer Perceptron (MLP) model. However, this approach yielded suboptimal results, indicating that the MLP model struggled to capture the complex patterns within the HPC data due to its limited ability to extract hierarchical features from raw inputs.

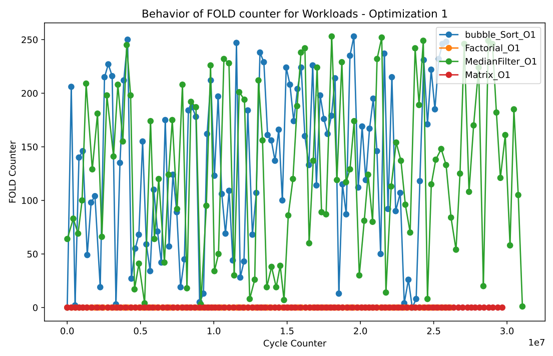

Fig. 4: Folded-Instruction Counter Behavior

To address this limitation, we employed a Convolutional Neural Network (CNN) model on HPC data for both workload and optimization identification. The CNN's ability to automatically learn spatial hierarchies of features made it a suitable candidate for handling the intricacies of HPC data. While the HPC data alone was insufficient for the CNN model to accurately identify both workloads and optimizations simultaneously—possibly due to the limited granularity since these counters are only limited to 8-bit and sampling rate inherent in the HPC measurements—focusing solely on workload identification produced promising results. The CNN model demonstrated good performance in distinguishing between different workloads based on HPC data. These findings suggest that HPCs hold significant potential for workload identification in embedded systems, motivating their use in our work despite the challenges encountered in broader optimization identification. These results are obtained using a dataset of 120 HPC samples, divided into 80% for training and 20% for validation.

Recognizing the need for more detailed information to distinguish optimization levels, we incorporated power consumption data, which provides cycle-accurate measurements and reflects the processor's dynamic behavior more precisely. As shown in Table 1, feeding the power consumption dataset into the CNN model yielded significantly better results. For both workload and optimization detection, the CNN was able to identify even subtle changes introduced by different optimization levels, achieving an accuracy of over 90%. For workload detection alone, the results reached 100% accuracy, demonstrating that the combination of CNN and power consumption data is highly effective for both workload and optimization detection. These results are obtained using a dataset of 240 power consumption trace, divided into 80% for training and 20% for validation.

These results highlight the significant sensitivity of power consumption in capturing the nuances associated with various optimization applied to workloads. The cycle-accurate of the power measurements provides a richer and more detailed dataset for the CNN to learn from, enabling it to discern patterns that were not evident from HPC data alone. This underscores the importance of integrating fine-grained power consumption with neural network models to enhance the identification and analysis of workloads and their optimization levels.

Table 1: Classification Results Leveraging Power Consumption and HPC Data

Power Consumption Based Detector Validation Accuracy		
Proposed Model	**MLP**	**CNN**
Workload and Optimization	55 %	91.67%
Workload Only	70 %	100 %
HPC Based Detector Validation accuracy (4 Counters)		
Proposed Model	**MLP**	**CNN**
Workload and Optimization	33 %	41.67 %
Workload Only	50 %	97.22 %

Throughout our research, we encountered several limitations. A major challenge was the low sampling rate of the debugger, which adversely affected the learning accuracy of our HPC based models due to insufficient temporal resolution in the collected data. In our system, these counters were encoded in 8 bits, causing them to reset to zero after reaching a maximum value of 255. This overflow limited the precision and reliability of the HPC data, We believe that even with the inclusion of additional HPCs, this limitation will continue to hinder the model's learning.. Employing a cycle-accurate debugger with higher-resolution counters could improve data collection precision, thereby enhancing model accuracy.

In contrast, the analysis of power consumption was more straightforward and yielded clearer insights. At each optimization level and across different workloads, we observed distinct impacts on the power consumption patterns. Therefore, CNN is able to achieve good results, with the ability to accurately detect both workloads and optimization. The results show how compiler optimizations can impact detection accuracy, as the CNN is able to detect the workload at 97.22%. In parallel, and for the same workload, the model is not able to detect the workload and its order of optimization, achieving an accuracy of about 41.67%. Previous work proved that HPC systems are good at profiling applications, while the optimizations induced by the compiler introduce uncertainty, change the identity of the workload, and mislead the model's detection, resulting in incorrect workload identification. However, power consumption is still able to detect the workload with its different levels of optimization, though at the cost of a high sampling rate, which should be at least equal to the processor speed.

5 Conclusion and Future work

This paper presented a comparative analysis of HPCs and power consumption as means for identifying and characterizing the behavior of embedded systems.

Through a series of experiments on ARM Cortex-M4 processors, we demonstrated that while HPCs can provide useful low-level insights into workload execution, their effectiveness is often limited by sampling rate and counter overflow. In contrast, power consumption measurements—captured at a cycle-accurate granularity—showed a superior ability to reflect both workload identity and optimization levels, achieving classification accuracy of about 91.67%.

Our findings underscore the potential of power consumption as a rich, non-intrusive behavioral fingerprint in embedded systems. Unlike HPCs, power traces can be collected externally with minimal interference, making them highly suitable for real-time monitoring in resource-constrained environments. Relying on the accuracy of power consumption, this can lead us to detect any deviation from the baseline which will help in malware detection in such resource-constrained embedded systems. Finally, our study opens several promising research avenues within hardware-assisted security and anomaly detection, particularly in the context of zero-day malware and firmware manipulation.

References

1. Aslan, O.A., Samet, R.: A comprehensive review on malware detection approaches. IEEE Access **8**, 6249–6271 (2020). https://doi.org/10.1109/ACCESS.2019.2963724
2. Azmoodeh, A., Dehghantanha, A., Conti, M., Choo, K.K.R.: Detecting crypto-ransomware in iot networks based on energy consumption footprint. Journal of Ambient Intelligence and Humanized Computing **9** (08 2018). https://doi.org/10.1007/s12652-017-0558-5
3. Basu, K., Krishnamurthy, P., Khorrami, F., Karri, R.: A theoretical study of hardware performance counters-based malware detection. IEEE Transactions on Information Forensics and Security **15**, 512–525 (2020). https://doi.org/10.1109/TIFS.2019.2924549
4. Bridges, R., Hernández Jiménez, J., Nichols, J., Goseva-Popstojanova, K., Prowell, S.: Towards malware detection via cpu power consumption: Data collection design and analytics. In: 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). pp. 1680–1684 (2018). https://doi.org/10.1109/TrustCom/BigDataSE.2018.00250
5. Celdrán, A.H., Sánchez, P.M.S., Castillo, M.A., Bovet, G., Pérez, G.M., Stiller, B.: Intelligent and behavioral-based detection of malware in iot spectrum sensors. Int. J. Inf. Secur. **22**(3), 541–561 (Jul 2022). https://doi.org/10.1007/s10207-022-00602-w, https://doi.org/10.1007/s10207-022-00602-w
6. He, Z., Makrani, H.M., Rafatirad, S., Homayoun, H., Sayadi, H.: Breakthrough to adaptive and cost-aware hardware-assisted zero-day malware detection: A reinforcement learning-based approach. In: 2022 IEEE 40th International Conference on Computer Design (ICCD). pp. 231–238 (2022). https://doi.org/10.1109/ICCD56317.2022.00042
7. Hernandez Jimenez, J., Goseva-Popstojanova, K.: Malware detection using power consumption and network traffic data. In: 2019 2nd International Conference on Data Intelligence and Security (ICDIS). pp. 53–59 (2019). https://doi.org/10.1109/ICDIS.2019.00016

8. Karnouskos, S.: Stuxnet worm impact on industrial cyber-physical system security. In: IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society. pp. 4490–4494 (2011). https://doi.org/10.1109/IECON.2011.6120048
9. Kasarapu, S., Shukla, S., Hassan, R., Sasan, A., Homayoun, H., Dinakarrao, S.M.P.: Generative ai-based effective malware detection for embedded computing systems (2024), https://arxiv.org/abs/2404.02344
10. Leng, E.W.L., Zwolinski, M., Halak, B.: Hardware performance counters for system reliability monitoring. In: 2017 IEEE 2nd International Verification and Security Workshop (IVSW). pp. 76–81 (2017). https://doi.org/10.1109/IVSW.2017.8031548
11. Makrakis, G.M., Kolias, C., Kambourakis, G., Rieger, C., Benjamin, J.: Industrial and critical infrastructure security: Technical analysis of real-life security incidents. IEEE Access **9**, 165295–165325 (2021). https://doi.org/10.1109/ACCESS.2021.3133348
12. Pan, Z., Sheldon, J., Sudusinghe, C., Charles, S., Mishra, P.: Hardware-assisted malware detection using machine learning. In: 2021 Design, Automation and Test in Europe Conference and Exhibition (DATE). pp. 1775–1780 (2021). https://doi.org/10.23919/DATE51398.2021.9474050
13. Patel, N., Sasan, A., Homayoun, H.: Analyzing hardware based malware detectors. In: Proceedings of the 54th Annual Design Automation Conference 2017. DAC '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3061639.3062202, https://doi.org/10.1145/3061639.3062202
14. Pham, D.P., Marion, D., Mastio, M., Heuser, A.: Obfuscation revealed: Leveraging electromagnetic signals for obfuscated malware classification. In: Proceedings of the 37th Annual Computer Security Applications Conference. p. 706–719. ACSAC '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3485832.3485894, https://doi.org/10.1145/3485832.3485894
15. Salehi, M., Hughes, D., Crispo, B.: µsbs: Static binary sanitization of bare-metal embedded devices for fault observability (10 2020)
16. Sharma, C., Moylan, S., Vasserman, E.Y., Amariucai, G.T.: Review of the security of backward-compatible automotive inter-ecu communication. IEEE Access **9**, 114854–114869 (2021). https://doi.org/10.1109/ACCESS.2021.3104854
17. Walker, A., Sengupta, S.: Malware family fingerprinting through behavioral analysis. In: 2020 IEEE International Conference on Intelligence and Security Informatics (ISI). pp. 1–5 (2020). https://doi.org/10.1109/ISI49825.2020.9280529
18. Yang, H., Tang, R.: Power consumption based android malware detection. Journal of Electrical and Computer Engineering **2016**, 1–6 (01 2016). https://doi.org/10.1155/2016/6860217
19. Zhou, B., Gupta, A., Jahanshahi, R., Egele, M., Joshi, A.: Hardware performance counters can detect malware: Myth or fact? In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security. p. 457–468. ASIACCS '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3196494.3196515, https://doi.org/10.1145/3196494.3196515