Special Session: Security and RAS in the Computing Continuum

Security and RAS in the Computing Continuum

Martí Alonso[1], David Andreu[1], Ramon Canal[1], Stefano Di Carlo[2], Odysseas Chatzopoulos[3], Cristiano Chenet[2], Juanjo Costa[1], Andreu Girones[1], Dimitris Gizopoulos[3], George Papadimitriou[3], Enric Morancho[1], Beatriz Otero[1], Alessandro Savino[2]

[1]Universitat Politècnica de Catalunya, Barcelona, Spain
[2]Politecnico di Torino, Torino, Italy
[3]University of Athens, Athens, Greece
Contact email: ramon.canal@upc.edu

***Abstract*—Security and RAS are two non-functional requirements under focus for current systems developed for the computing continuum. Due to the increased number of interconnected computer systems across the continuum, security becomes especially pervasive at all levels, from the smallest edge device to the high-performance cloud at the other end. Similarly, RAS (Reliability, Availability, and Serviceability) ensures the robustness of a system towards hardware defects. Namely, making them reliable, with high availability and design for easy service.**

In this paper and as a result of the Vitamin-V EU project, the authors detail the comprehensive approach to malware and hardware attack detection; as well as, the RAS features envisioned for future systems across the computing continuum.

***Index Terms*—RISC-V, Security, Malware, Hardware Attack, Computing Continuum, Simulation**

I. Introduction

RISC-V is a revolutionary open-source instruction set architecture (ISA) designed to offer simplicity, modularity, and extensibility [1]. This exciting development brings many benefits over proprietary processor architectures, including the potential for customization and lower licensing costs [2].

Despite these advantages and the fact that RISC-V applications have started to see their birth in the embedded domain [3], several challenges still need to be addressed before RISC-V can be widely adopted beyond conventional ones like performance or standardization. Among these challenges, security and RAS are of utmost importance. As RISC-V processors become more widely adopted, there is an increasing potential for security attacks. Ensuring the security of RISC-V-based applications will be an important challenge that needs to be addressed as technology develops.

Another key challenge is RAS. Enterprise and cloud data centers are increasingly integrating complex System-on-Chip (SoC) architectures to meet the demands of modern computing workloads. However, the widespread deployment of these devices raises the risk of undetected faults, which can lead to critical issues like system crashes or silent data corruptions (SDCs). These faults, stemming from manufacturing defects and in-field reliability concerns, present significant challenges for data centers. While cosmic ray-induced soft errors have been extensively studied [4]–[11], modern data centers must also consider other potential fault sources, such as manufacturing defects and thermal variations [12], which may lead to SDCs during normal operations [13]–[17].

This paper presents AI-based malware and hardware attack detection. With special emphasis on the reproducible and cross-platform methodology; as well as RAS.

The paper is organized as follows: Section II describes the step-by-step methodology used for AI-based security and Section III provides it for RAS. Then, Section IV presents the performance and detection capabilities of the AI-based security and the RAS analysis. Finally, Section V summarizes the main contributions of the paper.

II. Methodology for AI-based security

This section describes the methodology for the AI-based malware and hardware attack detection. We propose two different sources of information to detect the malign behaviour: hardware performance monitors (HPM) and instruction opcodes. While hardware performance monitors are only available during the execution of the program (thus, dynamically), opcodes are both available statically and dynamically. Consequently, in this section we describe the process of using each of these sources of information to process through ML algorithms to detect the attacks.

A. Hardware Performance Monitoring

Modern processors include hardware performance monitoring units to track the CPU performance, necessary due to increased processor complexity, such as hierarchical cache subsystems, non-uniform memory, and out-of-order execution. Software that adapts to resource utilization benefits from better performance and efficiency. The HPM includes registers and counters for microarchitectural events accessible by the Operating System (OS) using libraries like Linux perf [18] and PAPI [19].

HPMs track events such as retired instructions, branch predictions, cache hits/misses, and clock ticks, but only a limited

Funded by the European Union under the Horizon Europe Programme. Project name: Vitamin-V. Project number: 101093062. Views and opinions expressed are, however, those of the authors only and do not necessarily reflect those of the European Union or the HaDEA. Neither the European Union nor the granting authority can be held responsible for them. This paper is also funded by HFRI with title REDESIGN and Project Number 16973 and project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

number of hardware counters can be active at a time due to design and cost constraints [20]–[22]. Developers access counters via performance monitoring instructions, reading, and writing counter values.

B. Static Analysis of applications

Evaluating software trustworthiness early is essential. We developed a machine learning (ML) tool that analyzes static executable content to determine if it is benign or malicious. Inspired by previous work on detecting software bugs and security threats using static analysis [23], [24], we incorporated deep learning (DL) techniques to identify complex patterns. In cases where the dataset is insufficient or a zero-day attack is involved, transfer learning (TL) can be used to enhance detection accuracy by leveraging pre-trained models [25].

Following Haddadpajouh et al. [24] approach, we breakdown the process in four steps: dataset creation, feature extraction, model training, and deployment. To train the model, we gather a balanced dataset of benign and malicious programs, including hardware attacks like Spectre [26], Meltdown [27], viruses, and malware. Malicious programs will be sourced from platforms like VirusTotal [28], VirusShare [29], or SourceFinder [30]. Benign programs will come from regular Linux applications in the Debian repository.

Programs are converted to feature vectors by disassembling ELF binaries to extract operation codes (OpCode) sequences. These sequences are analyzed to generate feature vectors to train the DL model. The model developed and benchmarked in Section IV uses AMD64. Yet, the same methodology can be applied to other ISAs.

C. Dynamic Analysis using Hardware Performance Counters

Anomaly detection using hardware monitoring and AI involves dynamic analysis of microarchitectural events via ML algorithms to identify abnormal behavior. This approach, first introduced by Demme et al. in 2013 [31], has been applied to malware detection [32] and hard and soft errors [33], [34] but not on RISC-V.

Programs exhibit phase behaviors [35], [36], allowing for anomaly detection through patterns in hardware performance counters. The proposed anomaly detection framework includes three main components: (i) a CPU with HPM, (ii) data collection, and (iii) anomaly detection via ML classifiers. Challenges arise due to the cost and complexity of monitoring speculative execution events, especially in resource-constrained devices, where balancing hardware events and detection accuracy is critical. To overcome limited counter availability, some methods run applications multiple times to capture more events [31], [37], [38], but this impacts runtime applicability.

Data collection involves selecting events, extracting features, and reducing dimensions [32]. Feature extraction captures HPCs into vector space, while dimensionality reduction minimizes redundant data that can decrease detection accuracy. When empirical event selection is not feasible, techniques like Principal Component Analysis (PCA), Fisher Score, and Information Gain are used to identify relevant features [39]. The final block, anomaly detection, is carried out by ML classifiers. These classifiers can vary by type (multi-class or one-class), learning method (supervised, unsupervised, semi-supervised), or underlying algorithm (Neural Networks, Decision Trees, etc.). Multi-class classifiers assume multiple normal classes, while one-class classifiers identify anomalies based on a single normal class boundary [40]. Given the challenges of labeling data, unsupervised learning is often preferred [40]. To improve accuracy, advanced methods like Ensemble Learning [41], Boosting [42], and multi-stage classifiers [43] are used.

III. Methodology for RAS

A. Reliability, Availability, and Serviceability in Large-Scale SoC Deployment

Detecting and managing faults that result in SDCs is particularly challenging due to the specific conditions required for them to manifest [44]–[46]. These conditions can include particular machine instruction sequences, variations in operating voltages [47], temperature fluctuations, and platform-level behaviors like interrupts. This complexity results in low repeatability in SDC testing, necessitating extended testing periods to uncover potential issues. Developing effective testing methodologies to identify and address SDCs is therefore crucial for maintaining reliability, availability, and serviceability (RAS) in data centers. These strategies may involve repeated execution of specific code sequences to trigger SDCs or the use of pseudo-random instruction sequences to increase variability and expose latent faults during testing [48].

B. Sources of Faults and Impact on Reliability

SoCs can experience faults from various sources, including radiation, electrical marginalities, and silicon defects that may not be detectable during manufacturing. These issues may manifest in the field, impacting system reliability. The effects of such faults vary depending on where they occur within the SoC circuitry. For example, faults in error detection and correction-equipped circuits, such as caches, can be corrected at the hardware level, preventing system disruption [49]. However, SDCs where data errors propagate without triggering an interruption can have unpredictable consequences depending on the application [50]. A minor data error in a graphical operation might be negligible, but the same issue in a financial transaction could have serious implications.

Managing SDCs at scale is particularly crucial when deploying millions of processing cores, as even a single defect can cause significant operational disruptions. For instance, a modest-sized data center with 100,000 SoCs could experience at least one SDC per month at a 10 failures in time (FIT) rate [13] (where 1 FIT corresponds to one failure per billion hours of operation) [46], [48]. This occurrence rate underscores the need for rigorous reliability, availability, and serviceability (RAS) strategies to mitigate the effects of SDCs in large-scale computing environments. Hyperscale data centers, with millions of deployed SoCs, face a more acute risk, making reliability assessments and fault management an ongoing priority.

C. RAS-Oriented Approach

Ensuring the reliability, availability, and serviceability of data center infrastructure requires more than simply detecting data corruption and defective chips. It demands a comprehensive strategy that encompasses both hardware and software design. Reliability can be enhanced by identifying and isolating defective components through regular testing, while availability ensures systems continue to function smoothly even in the presence of faults. Serviceability emphasizes quick fault diagnosis and repair to minimize downtime and provide useful information for system debug and defect root causing.

One effective strategy for early-stage mitigation is simulation-based testing. Researchers can use architectural and microarchitectural models to simulate SoC behavior under various conditions, identifying potential faults before physical chips are manufactured. These models allow prediction of FIT rates early in the design process, enabling adjustments when they are less costly. However, early-stage models only approximate final hardware behavior, as they often lack complete design details.

In the late stages of development, more precise measurements can be made using gate-level models and actual silicon. However, these methods are more resource-intensive and time-consuming, limiting their applicability to scenarios where extremely high accuracy is required, such as in critical systems or hyperscale environments. By combining early predictive models with late-stage testing, data centers can implement a robust RAS framework to proactively address SDC risks.

IV. Results

A. AI-based security

The preliminary results on detecting malware and hardware attacks using HPCs are presented in this section, leveraging both supervised and unsupervised learning classifiers.

1) Supervised learning: With supervised classifiers, the detection of a specific type of hardware attack, i.e., side-channel attacks, is analyzed during runtime by monitoring HPCs.

Dataset creation: We selected 7 hardware-based attacks (i.e., Meltdown [27], Spectre [26](V1, V2 and V4), ZombieLoad [51], Fallout [52] and Crosstalk [53]) and 7 benign programs (i.e., Matrix multiplier, Debian stress tool [54], MiBench Bitcount [55], STREAM benchmark [56], bzip2 [57] and FFmpeg [58]) to construct the dataset.

We executed these applications and recorded multiple HPCs during binary execution using the *perf* tool. To streamline operations, we restricted execution to a single CPU core using the `taskset` Linux tool, ensuring that the collected HPC data remains unaffected by workload distribution across cores.

Some hardware attacks, like those in the Spectre family, exploit wrong speculative execution, triggered when the branch predictor mispredicts the outcome of a branch instruction. Therefore we selected both `branch-instructions` and `branch-misses` generic *perf* events, which provide the ratio between total branches and those where the predictor missed. This selection is supported by previous work, such as Congmiago Li et al. [59].

TABLE I
SAMPLES DISTRIBUTION FOR THE 3 SCENARIOS: BALANCED, ONLY MALIGN, AND ONLY BENIGN.

Dataset	Train		Test	
	Malign	Benign	Malign	Benign
Balanced	11200	11200	2800	2800
Malign	11200	0	2800	14000
Benign	0	11200	14000	2800

Additionally, side-channel hardware-based attacks heavily stress the computer's cache memory. A high count of cache misses on the last-level cache (LLC) memory may indicate the presence of a FLUSH+RELOAD side-channel attack, known as the most effective and popular among hardware-based attacks. The first-level cache is also a common target in other attacks, as used by Stefano Carnà et al. [60]. Thus, we also selected the `LLC-load-misses` and `L1-dcache-load-misses` events.

Previous works have used sampling rates ranging from 1 ms per sample to 100 ms per sample. Congmiago Li et al. [59] even dynamically change the sample rate to prevent evasive malware. To generate a large number of samples for the machine learning model, the aim was to use the lowest possible sample rate. However, experimental results showed that rates below 10 ms caused anomalies in *perf*, such as missed samples. As a result, we chose a 1 ms sample rate. We extract 2,000 samples from each application, either by running the application for 2 seconds or repeating the execution until we get those samples.

Dataset preparation: The dataset for the 14 applications contains 28,000 samples, we created 3 different scenarios: Balanced, Only malign, and Only benign. Each scenario was split using 80% for training and 20% for testing (as shown in Table I).

Model evaluation: We used different models for training: Support Vector Machine (SVM) and One-Class SVM. The SVM model shows the best results for detecting side-channel attacks with a 99% accuracy, where only 2 of the attacks were misdetected (as observed in the confusion matrix of Figure 1 left).

Figure 1 (center and right) shows the scenarios where a single class dataset is used for training (either benign or malign). In these cases, we observed many false positives because the model encounters samples that were unseen during training and therefore misclassifies them as malware.

2) Unsupervised learning: To evaluate the effectiveness of unsupervised learning techniques, an experiment was designed to detect Stack Buffer Overflow (SBO) attacks. Four applications from the MiBench suite were selected—AES, RSA, SHA, and Dijkstra—and were subjected to SBO attacks. These attacks triggered the execution of a malicious function, with its size parametrized relative to the original code (i.e., the smaller the malicious function, the more stealthy its execution).

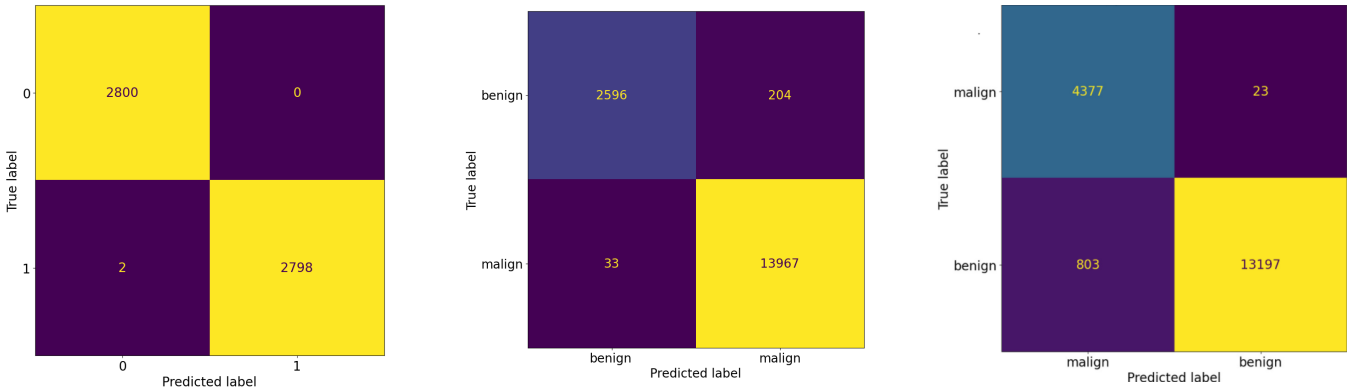

Fig. 1. Confusion matrices for detecting malware using a balanced data set (left), a benign dataset (center) and a malign dataset (right).

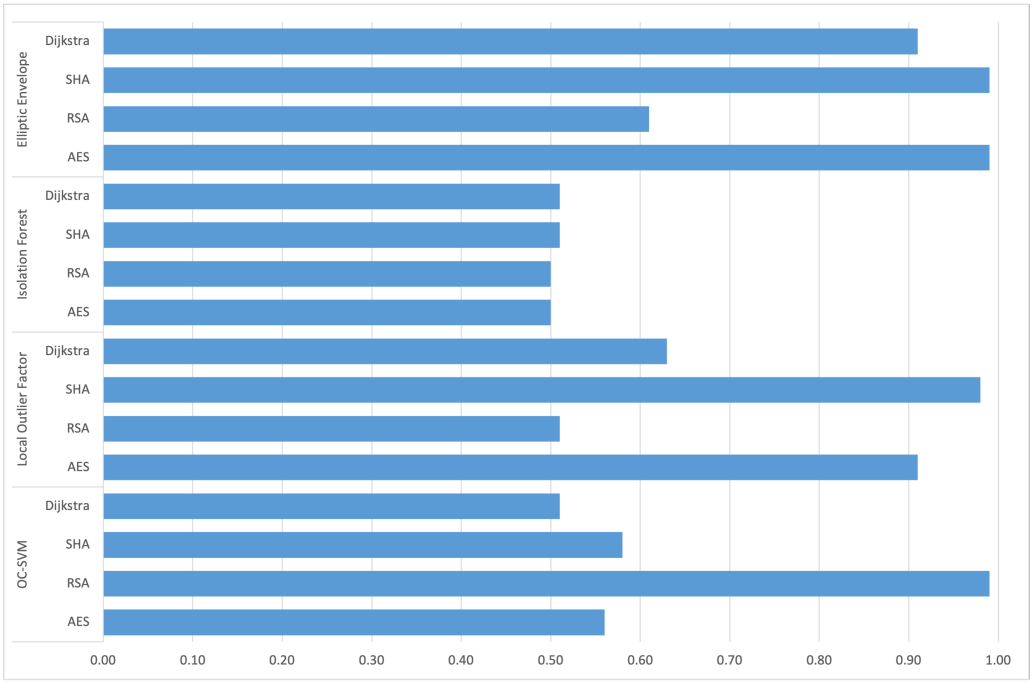

Fig. 2. Accuracy of unsupervised SBO detection for different benchmarks and classifiers. The malicious function runs a number of instructions lower than 1% of those of the original application.).

For each application, 20,000 executions were collected (10,000 for training and 10,000 for testing), with benchmark inputs randomly varied. The training dataset consisted solely of benign executions, while the testing dataset included 50% benign executions and 50% with attacks. Four anomaly detection models—OC-SVM, LOF, Isolation Forest, and Elliptic Envelope—were trained to assess their effectiveness on the dataset. Preliminary results, shown in Figure 2, indicate promising performance for the Elliptic Envelope classifier, except the RSA benchmark, which remains under investigation.

TABLE II
MAJOR SIMULATOR CONFIGURATIONS FOR EACH ISA.

Parameter	Value
ISA	RISC-V / Arm / x86
Pipeline	64-bit OoO (8-issue)
L1 Instruction Cache	32KB, 64B line, 128 sets, 4-way
L1 Data Cache	32KB, 64B line, 128 sets, 4-way
L2 Cache	1MB, 64B line, 2048 sets, 8-way
Physical Register File	128 Int; 128 FP
LQ/SQ/IQ/ROB entries	32/32/64/128

B. RAS

This subsection showcases the effects of *Permanent* faults on program execution. We focus on SDC outcomes, showing the probability that a fault in a specific hardware unit results in an SDC. The basic parameters of the gem5 configuration we use in this paper can be seen in Table II.

1) SDCs due to Permanent Faults in L1 instruction and data caches: Fig. 3 illustrates the SDC probability outcomes for permanent faults in the L1 Instruction Cache across fifteen

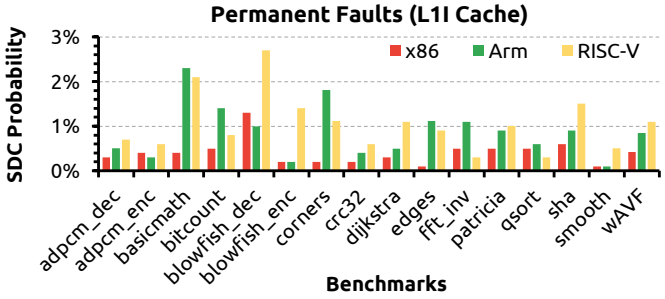

Fig. 3. SDC probability due to permanent faults in L1 instruction cache [46].

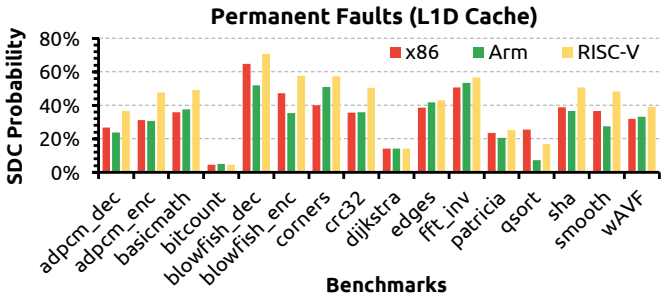

Fig. 4. SDC probability due to permanent faults in L1 data cache [46].

benchmarks of the MiBench [61] suite for the three ISAs (Arm, x86, RISC-V). As shown in Fig. 3, the SDC probability ranges from 0.1% to 2.3% for Arm ISA, 0.1% to 1.3% for x86, and 0.3% to 2.7% for RISC-V ISA. These results are expected because a workload running with a persistent fault in any level of cache memory that stores instructions is unlikely to survive to the end and produce a corrupted output. Faults in most fields of instruction will primarily impact the execution flow or the instruction operands, and thus, lead to a crash [10]. On average across all benchmarks, the x86 ISA demonstrates the lowest SDC probability among the ISAs studied in this paper, while the RISC-V ISA shows the highest SDC probability in most benchmarks.

Fig. 4 displays the SDC probability results for permanent faults in the L1 Data Cache across the same fifteen MiBench benchmarks for the three ISAs (Arm, x86, RISC-V). As shown in Fig. 4, the SDC probability varies from 5.1% to 53.3% for Arm ISA, 4.4% to 64.7% for x86, and 4.4% to 70.8% for RISC-V ISA. On average across all benchmarks, the RISC-V ISA exhibits the highest SDC probability among all ISAs studied in this paper. It is important to note that the L1 Data Cache is considered unprotected in our experiments, i.e., there is no ECC-related protection scheme. The actual SDC probability can be much lower in real systems due to these protection mechanisms.

Overall, for the microarchitecture and workloads analyzed, the RISC-V ISA demonstrates a significantly higher probability of SDCs due to permanent faults compared to the other ISAs, i.e., Arm and x86.

V. Conclusions

This paper provided an overview of AI-based security and RAS solutions in the computing continuum. The paper describes the comprehensive approach to malware and hardware attack detection; as well as, the RAS features envisioned for future systems across the computing continuum. AI-based detection is shown to be a highly effective way to detect malware either statically or dynamically and with several ML methods. The paper also provides an analysis of the vulnerability to SDCs of L1 data and instruction caches for the same core but different ISA. The results show the importance of RAS features in future systems as the vulnerability to SDCs increases in each technology.

References

[1] A. Waterman, Y. Lee *et al.*, "The RISC-V instruction set manual volume ii: Privileged architecture version 1.9," EECS Department, University of California, Berkeley, Tech. Rep., 2016.

[2] S. Greengard, "Will RISC-V revolutionize computing?" *Communications of the ACM*, vol. 63, no. 5, pp. 30–32, 2020.

[3] A. Dörflinger, M. Albers *et al.*, "A comparative survey of open-source application-class RISC-V processor implementations," in *Proceedings of the 18th ACM International Conference on Computing Frontiers*, ser. CF '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 12–20.

[4] G. Papadimitriou and D. Gizopoulos, "Anatomy of on-chip memory hardware fault effects across the layers," *IEEE Transactions on Emerging Topics in Computing*, vol. 11, no. 2, pp. 420–431, 2023.

[5] G. Papadimitriou and D. Gizopoulos, "Avgi: Microarchitecture-driven, fast and accurate vulnerability assessment," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023, pp. 935–948.

[6] G. Papadimitriou and D. Gizopoulos, "Demystifying the system vulnerability stack: Transient fault effects across the layers," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 902–915.

[7] A. Chatzidimitriou, G. Papadimitriou *et al.*, "Multi-bit upsets vulnerability analysis of modern microprocessors," in *2019 IEEE International Symposium on Workload Characterization (IISWC)*, 2019, pp. 119–130.

[8] P. Bodmann, G. Papadimitriou *et al.*, "The Impact of SoC Integration and OS Deployment on the Reliability of Arm Processors," in *2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2021, pp. 223–225.

[9] P. Bodmann, G. Papadimitriou *et al.*, "Impact of cores integration and operating system on arm processors reliability: Micro-architectural fault-injection vs beam experiments," in *2020 20th European Conference on Radiation and Its Effects on Components and Systems (RADECS)*, 2020, pp. 1–4.

[10] G. Papadimitriou and D. Gizopoulos, "Silent data corruptions: Microarchitectural perspectives," *IEEE Transactions on Computers*, vol. 72, no. 11, pp. 3072–3085, 2023.

[11] P. R. Bodmann, G. Papadimitriou *et al.*, "Soft error effects on arm microprocessors: Early estimations versus chip measurements," *IEEE Transactions on Computers*, vol. 71, no. 10, pp. 2358–2369, 2022.

[12] P. Koutsovasilis, C. D. Antonopoulos *et al.*, "The impact of cpu voltage margins on power-constrained execution," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 1, pp. 221–234, 2022.

[13] D. P. Lerner, B. Inkley *et al.*, "Optimization of tests for managing silicon defects in data centers," in *2022 IEEE International Test Conference (ITC)*, 2022, pp. 578–582.

[14] H. D. Dixit, S. Pendharkar *et al.*, "Silent Data Corruptions at Scale," 2021. [Online]. Available: https://arxiv.org/abs/2102.11245

[15] P. H. Hochschild, P. Turner *et al.*, "Cores That Don't Count," in *Proceedings of the Workshop on Hot Topics in Operating Systems*, ser. HotOS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 9–16.

[16] S. Wang, G. Zhang *et al.*, "Understanding silent data corruptions in a large production cpu population," in *Proceedings of the 29th Symposium on Operating Systems Principles*, ser. SOSP '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 216–230.

[17] D. Gizopoulos, "Sdcs: A b c," Sep 2024. [Online]. Available: https://www.sigarch.org/sdcs-a-b-c/

[18] J. M. Domingos, P. Tomas, and L. Sousa, "Supporting RISC-V performance counters through performance analysis tools for linux (perf)," in *5th Workshop on Computer Architecture Research with RISC-V (CARRV '21)*, 2021, pp. 935–948. [Online]. Available: https://arxiv.org/pdf/2112.11767.pdf

[19] S. Browne, J. Dongarra *et al.*, "A scalable cross-platform infrastructure for application performance tuning using hardware counters," in *SC '00: Proceedings of the 2000 ACM/IEEE Conference on Supercomputing*, 2000, pp. 42–42.

[20] B. Sprunt, "The basics of performance-monitoring hardware," *IEEE Micro*, vol. 22, no. 4, pp. 64–71, 2002.

[21] C. Malone, M. Zahran, and R. Karri, "Are hardware performance counters a cost effective way for integrity checking of programs," in *Proceedings of the Sixth ACM Workshop on Scalable Trusted Computing*, ser. STC '11. New York, NY, USA: Association for Computing Machinery, 2011, pp. 71–76.

[22] N. C. Doyle, E. Matthews *et al.*, "Performance impacts and limitations of hardware memory access trace collection," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, 2017, pp. 506–511.

[23] Y. Ding, W. Dai *et al.*, "Control flow-based opcode behavior analysis for malware detection," *Computers & Security*, vol. 44, pp. 65–74, 2014.

[24] H. HaddadPajouh, A. Dehghantanha *et al.*, "A deep recurrent neural network based approach for internet of things malware threat hunting," *Future Generation Computer Systems*, vol. 85, pp. 88–96, 2018.

[25] E. Rodríguez, P. Valls *et al.*, "Transfer-learning-based intrusion detection framework in iot networks," *Sensors*, vol. 22, no. 15, p. 5621, 2022.

[26] P. Kocher, J. Horn *et al.*, "Spectre attacks: Exploiting speculative execution," in *40th IEEE Symposium on Security and Privacy (S&P'19)*, 2019.

[27] M. Lipp, M. Schwarz *et al.*, "Meltdown: Reading kernel memory from user space," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018.

[28] "VirusTotal," https://www.virustotal.com/.

[29] "VirusShare," https://www.virusshare.com/.

[30] M. O. F. Rokon, R. Islam *et al.*, "{SourceFinder}: Finding malware {Source-Code} from publicly available repositories in {GitHub}," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, pp. 149–163.

[31] J. Demme, M. Maycock *et al.*, "On the feasibility of online malware detection with performance counters," *SIGARCH Comput. Archit. News*, vol. 41, no. 3, pp. 559–570, jun 2013.

[32] C. P. Chenet, A. Savino, and S. Di Carlo, "A survey on hardware-based malware detection approaches," *IEEE Access*, vol. 12, pp. 54 115–54 128, 2024.

[33] 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.

[34] D. Kasap, A. Carpegna *et al.*, "Micro-architectural features as soft-error induced fault executions markers in embedded safety-critical systems: a preliminary study," 2023.

[35] T. Sherwood, E. Perelman *et al.*, "Discovering and exploiting program phases," *IEEE Micro*, vol. 23, no. 6, pp. 84–93, 2003.

[36] C. Isci, G. Contreras, and M. Martonosi, "Live, runtime phase monitoring and prediction on real systems with application to dynamic power management," in *2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)*, 2006, pp. 359–370.

[37] B. Singh, D. Evtyushkin *et al.*, "On the detection of kernel-level rootkits using hardware performance counters," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 483–493.

[38] H. Sayadi, N. Patel *et al.*, "Machine learning-based approaches for energy-efficiency prediction and scheduling in composite cores architectures," in *2017 IEEE International Conference on Computer Design (ICCD)*, 2017, pp. 129–136.

[39] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[40] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, jul 2009.

[41] H. Sayadi, N. Patel *et al.*, "Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 2018, pp. 1–6.

[42] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[43] H. Sayadi, H. M. Makrani *et al.*, "2smart: A two-stage machine learning-based approach for run-time specialized hardware-assisted malware detection," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2019, pp. 728–733.

[44] D. Gizopoulos, G. Papadimitriou, and O. Chatzopoulos, "Estimating the failures and silent errors rates of cpus across isas and microarchitectures," in *2023 IEEE International Test Conference (ITC)*, 2023, pp. 377–382.

[45] T. Macieira, S. Gurumurthy *et al.*, "Silent data corruptions in computing: Understand and quantify," in *2024 IEEE 30th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2024, pp. 1–7.

[46] D. Gizopoulos, G. Papadimitriou *et al.*, "Silent data corruptions in computing systems: Early predictions and large-scale measurements," in *2024 IEEE European Test Symposium (ETS)*, 2024, pp. 1–10.

[47] G. Papadimitriou, M. Kaliorakis *et al.*, "A system-level voltage/frequency scaling characterization framework for multicore cpus," in *IEEE Silicon Errors in Logic – System Effects (SELSE 2017)*, 2017.

[48] N. Karystinos, O. Chatzopoulos *et al.*, "Harpocrates: Breaking the silence of cpu faults through hardware-in-the-loop program generation," in *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, 2024, pp. 516–531.

[49] D. Agiakatsikas, G. Papadimitriou *et al.*, "Impact of voltage scaling on soft errors susceptibility of multicore server cpus," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 957–971.

[50] G. Papadimitriou, D. Gizopoulos *et al.*, "Silent data corruptions: The stealthy saboteurs of digital integrity," in *2023 IEEE 29th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2023, pp. 1–7.

[51] M. Schwarz, M. Lipp *et al.*, "ZombieLoad: Cross-privilege-boundary data sampling," in *CCS*, 2019.

[52] C. Canella, D. Genkin *et al.*, "Fallout: Leaking data on meltdown-resistant cpus," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2019.

[53] H. Ragab, A. Milburn *et al.*, "CrossTalk: Speculative Data Leaks Across Cores Are Real," in *S&P*, May 2021, intel Bounty Reward. [Online]. Available: https://download.vusec.net/papers/crosstalk_sp21.pdf

[54] R. O. S. Projects, "stress," https://github.com/resurrecting-open-source-projects/stress.

[55] U. of Michigan, "Mibench version 1.0," https://vhosts.eecs.umich.edu/mibench/, 2002, accessed: 14-05-2024.

[56] J. D. McCalpin, "Stream: Sustainable memory bandwidth in high performance computers," https://www.cs.virginia.edu/stream/, accessed: 28-05-2024.

[57] "bzip2," https://sourceware.org/bzip2/, accessed: 28-05-2024.

[58] "Ffmpeg," https://ffmpeg.org/, accessed: 28-05-2024.

[59] C. Li and J.-L. Gaudiot, "Detecting spectre attacks using hardware performance counters," *IEEE Transactions on Computers*, vol. 71, no. 6, pp. 1320–1331, 2022.

[60] S. Carnà, S. Ferracci *et al.*, "Fight hardware with hardware: Systemwide detection and mitigation of side-channel attacks using performance counters," *Digital Threats*, vol. 4, no. 1, 3 2023.

[61] M. R. Guthaus, J. S. Ringenberg *et al.*, "Mibench: A free, commercially representative embedded benchmark suite," in *Proceedings of the fourth annual IEEE international workshop on workload characterization. WWC-4 (Cat. No. 01EX538)*. IEEE, 2001, pp. 3–14.