Validation, Verification, and Testing (VVT) of future RISC-V powered cloud infrastructures: the Vitamin-V Horizon Europe Project perspective

Martí Alonso[1], David Andreu[1], Ramon Canal[1], Stefano Di Carlo[2], Cristiano Chenet[2], Juanjo Costa[1], Andreu Girones[1], Dimitris Gizopoulos[3], Vasileios Karakostas[3], Beatriz Otero [1], George Papadimitriou[3], Eva Rodríguez[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: stefano.dicarlo@polito.it

***Abstract*—Vitamin-V is a project funded under the Horizon Europe program for the period 2023-2025. The project aims to create a complete open-source software stack for RISC-V that can be used for cloud services. This software stack is intended to have the same level of performance as the x86 architecture, which is currently dominant in the cloud computing industry. In addition, the project aims to create a powerful virtual execution environment that can be used for software development, validation, verification, and testing. The virtual environment will consider the relevant RISC-V ISA extensions required for cloud deployment. Commercial cloud systems use hardware features currently unavailable in RISC-V virtual environments, including virtualization, cryptography, and vectorization. To address this, Vitamin-V will support these features in three virtual environments: QEMU, gem5, and cloud-FPGA prototype platforms. The project will focus on providing support for EPI-based RISC-V designs for both the main CPUs and cloud-important accelerators, such as memory compression. The project will add the compiler (LLVM-based) and toolchain support for the ISA extensions. Moreover, Vitamin-V will develop novel approaches for validating, verifying, and testing software trustworthiness. This paper focuses on the plans and visions that the Vitamin-V project has to support validation, verification, and testing for cloud applications, particularly emphasizing the hardware support that will be provided.**

***Index Terms*—RISC-V, Validation, Verification, Testing, Cloud computing, 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 for cloud applications. One key obstacle is the maturity of the RISC-V ecosystem. The platform has gained significant momentum in recent years, but the ecosystem surrounding RISC-V processors is still developing. This includes hardware and software tools and the number of vendors and support services available [4]. As the ecosystem continues to mature, it is expected that this will become less of a concern.

Another potential challenge is performance. Although RISC-V processors can offer good performance, they may not yet be able to match the performance of more established architectures, such as x86 or ARM, in specific applications. This could limit the adoption of RISC-V in performance-sensitive cloud applications. As technology continues to evolve, this may become less of a barrier.

Compatibility is another potential challenge to widespread RISC-V adoption. Many cloud applications are designed to run on x86 or ARM architectures and may not be compatible with RISC-V processors. This could limit the use of RISC-V in specific cloud environments. Efforts are underway to address this issue by developing emulation and virtualization solutions.

Security is also a concern. As RISC-V processors become more widely adopted, there is an increasing potential for security attacks. Ensuring the security of RISC-V-based cloud applications will be an important challenge that needs to be addressed as technology develops.

Eventually, standardization is another area that needs to be addressed. While RISC-V is an open standard, there is still a need for further standardization in areas such as memory management and I/O interfaces. This can lead to compatibility issues between different RISC-V implementations and limit the portability of RISC-V-based cloud applications. Efforts are underway to address this issue by developing standardization solutions that can promote interoperability.

Vitamin-V is a research project funded by the European Commission in the 2023-2025 time frame to propose innovative solutions that aim to address these challenges. Vitamin-

Funded by the European Union. 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. Project number: 101093062

 https://dx.doi.org/10.1109/ETS56758.2023.10174216

V will deploy a complete RISC-V hardware-software stack for cloud services based on cutting-edge cloud open-source technologies for RISC-V cores, with a focus on EPI cores. Vitamin-V is designed to offer an innovative RISC-V virtual execution environment, which provides hardware emulation, simulation, and FPGA prototyping to enable software development, verification, and validation before actual hardware is released. Additionally, Vitamin-V contributes to porting the complete cross-compiling toolchain, software stack, and essential application libraries for the forthcoming release of the RISC-V EPI processors [5]. This solution is expected to play a critical role in promoting the adoption of RISC-V for cloud applications [6].

In particular, Validation, Verification, and Testing (VVT) activities are among the most critical activities during software development and deployment, with potential risks in terms of the safety and security of cloud applications. After providing a general overview of the Vitamin-V activities, this paper wants to focus on plans and visions that the Vitamin-V project has to support VVT activities for cloud applications, concentrating on the support that the hardware itself can provide to these activities.

The paper is organized as follows: Section II overviews the Vitamin-V project organization, while Section III provides a deeper overview of the project's implementation plans VVT techniques for RISC-V cloud applications. Finally, Section IV summarizes the main contributions of the paper.

II. Concept and Methodologies

The objective of Vitamin-V is to create a complete RISC-V cloud software stack that can compete with the dominant x86 counterpart in terms of performance, as shown in Figure 1. However, the lack of full-fledged RISC-V systems presents a significant challenge for porting and evaluating advanced cloud setups and software stacks. Commercial cloud systems use hardware features partially available in RISC-V virtual environments and commercial hardware cores. These features include virtualization, cryptography, and vector extensions.

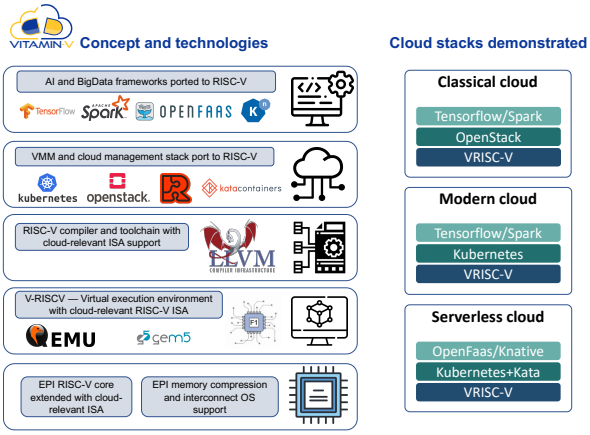

Fig. 1. VITAMIN-V concept and architecture

To address this challenge, Vitamin-V aims to develop the *Vitamin RISC-V virtual execution environment* (VRISC-V), which is a multi-layered high-performance RISC-V virtual environment based on three cutting-edge technologies: functional emulation (QEMU [7]), cycle-accurate simulation (gem5) [8], and an FPGA-based hardware prototype system node capable of running on AWS EC2 F1 FPGAs (and thus scalable to 100s of nodes) (FPGA [9]). These technologies provide unique features essential for software development, validation, verification, and testing, making it easier for software developers to adopt RISC-V. With QEMU and the RISC-V on FPGA, the project aims at simulating multi-node systems with more than 100 cores. The goal is to simulate multi-node systems in gem5 with cycle-accurate accuracy and microarchitectural configuration flexibility [10].

Vitamin-V will support accelerators, such as memory compression, in the VRISC-V virtual execution environment and provide a mature compiler toolchain based on LLVM [11] to handle the complete RISC-V ISA, including its extensions. Additionally, the project will develop a validation, verification, and testing (VVT) toolset to identify software bugs and malicious code sequences to ensure the trustworthiness of the software layers in future RISC-V systems. The VVT tools will use the VRISC-V platform to facilitate software porting and prototyping before mature hardware is available. This will promote the migration towards new RISC-V servers in the cloud.

To enable the execution of complete cloud stacks on the VRISC-V virtual execution environment, Vitamin-V will port all necessary machine-dependent modules in relevant open-source cloud software distributions. These modules include support for running entire Virtual Machines (VMs), containers, and lightweight VMs (KVM, QEMU, Docker, RustVMM), safety-security trusted execution environments (VOSySMonitoRV [12]), cloud management software (OpenStack, Kubernetes, Kata Containers), and AI and Big Data libraries (Tensorflow, Spark).

The project will address classic cloud stacks that target the execution of entire VMs managed by OpenStack, modern cloud setups that target entire VMs and containers managed by Kubernetes, and serverless cloud stacks that target the execution of lightweight VMs managed by Kubernetes with Kata Containers. Vitamin-V will benchmark the three working cloud setups against relevant AI applications (i.e., Google Net, ResNet, VGG19), Big-Data applications (TPC-DS on top of Apache Spark), and Serverless applications (FunctionBench, ServerlessBench). The goal is to match the software performance of its x86 equivalent, using CPU core mark scores for a fair assessment. Overall, this development will establish a RISC-V cloud-stack ecosystem for market adoption.

III. Validation, Verification, and Testing for RISC-V cloud services

In the age of cloud services, it is often difficult to transfer software workloads between different computing platforms without sacrificing performance and trustworthiness

[13]. Several factors can contribute to decreased performance when moving from one architecture to another, including poor software implementation, inefficient data structures, and limited use of caching. Additionally, the software can be compromised by bugs or malicious code at any point in its lifespan. As a result, to optimize and ensure the trustworthiness of an application, it is essential to monitor its execution, identify performance bottlenecks, and customize the software to fit the underlying hardware. However, this task cannot be accomplished using simple performance metrics such as execution time or clock cycles. Multiple factors come into play when mapping software to a modern computing system, including in- vs. out-of-order execution engines, pipeline stages, execution ports and corresponding latencies, re-order buffers, load/store queues, and cache organization. As a result, capturing and analyzing detailed performance metrics is becoming increasingly necessary to enable in-depth architecture modeling and optimization procedures [14].

This section explains how the Vitamin-V project intends to utilize the RISC-V Hardware Performance Monitor (HPM) unit to implement advanced Validation, Verification, and Testing for RISC-V cloud services.

A. Hardware Performance Monitoring

Like other modern processors, RISC-V processors come with hardware performance monitoring units to keep track of processor performance. These units have become necessary due to the increased complexity of processors in recent decades, which has resulted in the need for hierarchical cache subsystems, non-uniform memory, simultaneous multithreading, and out-of-order execution. Software that can understand and adjust to resource utilization has benefits for performance and efficiency.

The RISC-V ISA has a simpler HPM unit than x86, but it defines a flexible and open-source performance monitoring solution that can be implemented in various ways. Vitamin-V plans to support the latest RISC-V HPM specifications, starting with hardware simulation in VRISC-V and integrating it into the operating system using open-source libraries like Linux perf [15] and PAPI [16]. The implemented HPM will have hardware registers and counters for microarchitectural events that software can access and hardware assertions from gem5 useful for debugging [17]. Vitamin-V aims to build at least essential counter events during early development, using existing RISC-V cores like Semidynamics' Atrevido Core [18].

The hardware performance monitoring units track various events related to the processor's architecture and microarchitecture, such as retired instructions, branch predictions, cache hits and misses, floating-point operations, hardware interrupts, elapsed core clock ticks, and core frequency. These events generate several parameters, but the processors have only a few registers to store them, so only a few hardware performance counters are available at any given time. The design complexity and cost of concurrent monitoring of events limit the number of available hardware performance counters [19]–[21].

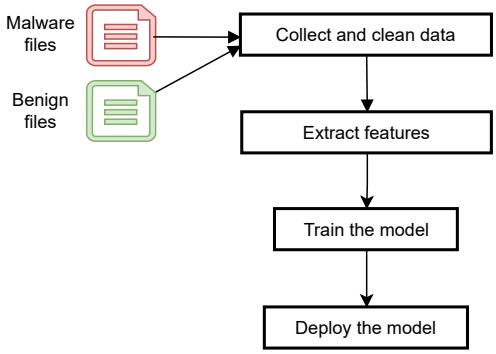

Fig. 2. Methodology used to create the Deep Learning model used in VITAMIN-V.

Performance monitoring instructions allow developers to access the counters by reading or writing their values. Vitamin-V will investigate VRISC-V's hardware monitor extensions to address the need for software validation, verification, and testing in cloud environments. The project plans to use hardware monitoring to assess the trustworthiness of software deployed on RISC-V. Vitamin-V will implement various monitoring events to support software validation, verification, and testing for future implementation in RISC-V cores, using the state-of-the-art gem5 simulator [8].

B. VVT through Static Analysis

It is crucial to evaluate the trustworthiness of software at an early stage before it is executed. Vitamin-V is planning to implement a machine learning (ML) tool to analyze the static content of executable files to determine whether they are benign or malicious. Previous work on using static analysis of executable files to detect software bugs and security threats [22], [23] has inspired Vitamin-V's decision to incorporate deep learning (DL) techniques to identify complex patterns from a vast amount of labeled data. However, there are situations where the dataset may not be sufficient or a zero-day attack detection is required. In such cases, a transfer learning (TL) technique can be implemented to enhance the accuracy of the detection process. Transfer learning involves reusing a pre-trained model for a different task [24].

The proposed methodology is based on a similar approach to the one used by Haddadpajouh et al. [23] and consists of four main steps as illustrated in Figure 2: dataset creation, feature extraction, model training, and model deployment and tuning.

The first step in training any neural network model is to have a balanced collection of benign and malicious codes. Vitamin-V focuses on detecting various types of malicious codes, including hardware attacks like spectre [25], meltdown [26], viruses, and other malware. To achieve this, Vitamin-V will gather different source codes from popular public repositories such as VirusTotal [27], VirusShare [28], or SourceFinder [29]. However, since there is currently a lack of source code

repositories for RISC-V malicious codes, the plan is to use the AMD64 version of the collected malware and recompile them for the target platform. This approach is suitable for the current situation, while new malicious codes for the RISC-V architecture can be added to the dataset when they become available. Vitamin-V plans to download a selection of common Linux application packages from the Debian repository, which already has RISC-V versions, for the benign codes. A balanced selection of both sets of codes will be built.

Each program must be represented as a feature vector to classify executables. Vitamin-V uses Linux as its operating system; therefore, the binary files use the Executable and Linking Format (ELF). These ELF binaries can be disassembled using default Linux tools to a textual format and extract their sequence of instructions (the operation codes or OpCodes). This sequence of instructions can be analyzed to obtain the frequency of different OpCodes sequences (the n-grams) and then calculate the feature vector for each sequence, which will help train the neural network.

The categorized feature vectors can be fed to the DL network and train it to generate a model capable of detecting benign or malicious codes. However, due to the previously mentioned lack of malicious codes in RISC-V, the resulting model will -initially- only be able to detect malicious code coming from AMD64 platforms.

To overcome this limitation, Vitamin-V plans to use transfer learning techniques to transfer this knowledge to a new neural network that will detect malicious code and zero-day attacks on RISC-V binaries. To fine-tune the model, new data will be collected, such as hardware performance counters, to adjust the parameters of the pre-trained model and improve its performance in detecting malware attacks. After an evaluation period to ensure the proper performance of the detection mechanism, the proposed model will be deployed. Additional data collection and fine-tuning steps may be required to keep the model accurate and effective over time.

C. Dynamic Analysis using Hardware Performance Counters

Although an application may be deemed trustworthy through static verification, corruption can still occur during runtime, necessitating dynamic monitoring. Such corruption can result from environmental conditions, electromagnetic fields, space radiation, aging, design flaws, manufacturing imperfections, and intentional attacks, which can generate abnormal behavior in processors, compromising the safety, reliability, and security of modern computing systems. Anomaly detection involves identifying patterns in data that deviate from expected behavior [30]. Vitamin-V intends to exploit the RISC-V HPM unit for this purpose, as demonstrated by the proof of concept reported in Figure 3, where the use of hardware counters distinguishes between benign and malicious applications. The figure shows a box plot of the frequency of a specific ratio, L3 misses per L1 misses, clearly identifying three malicious codes (MeltDown, Spectre, and ZombieLoad) in contrast to three benign applications (two applications from the MiBench benchmark and a YouTube video visualization in a Firefox browser).

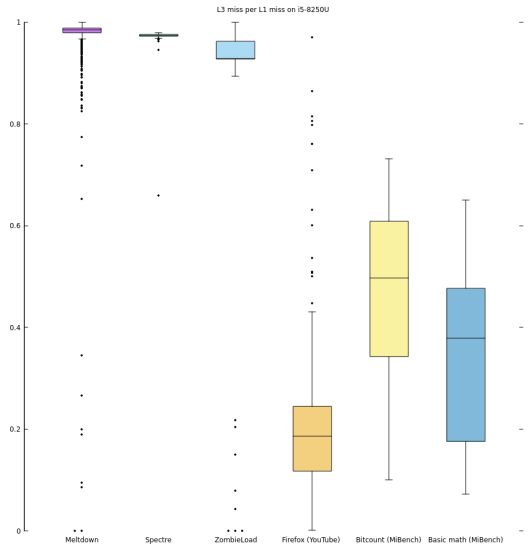

Fig. 3. Using hardware counters to detect benign and malicious code.

Anomaly detection through hardware and artificial intelligence (AI) involves the dynamic analysis of micro-architectural events in a processor by machine learning (ML) algorithms, which distinguish between normal and anomalous behavior. This approach was first introduced in the hardware-based malware detector proposed by Demme et al. in 2013 [31] and exploited for soft error detection in [32], [33].

The intuition behind detecting anomalies based on hardware performance counters (HPCs) stems from the fact that programs exhibit phase behavior [34], [35]. Programs perform activities in distinct phases, which can correspond to patterns in architectural and micro-architectural events, thus enabling the detection of anomalies based on time-behavioral patterns of the HPCs.

The Vitamin-V hardware and AI-based anomaly detector framework is presented in Figure 4, with its primary goal of differentiating between normal execution and anomalous execution. The framework comprises three fundamental blocks: (i) the RISC-V based processor with its HPCs, (ii) the data collection process, and (iii) the anomaly detection itself, performed by an ML classifier. The detector's performance and efficiency are evaluated to determine its effectiveness.

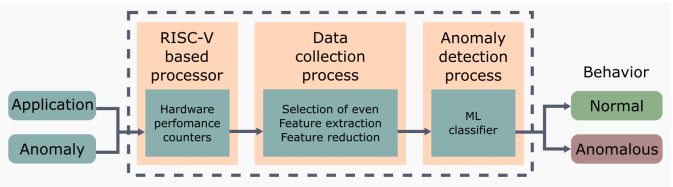

Fig. 4. The hardware and AI-based anomaly detector framework. Elaborated by the author.

Adding hardware performance counters presents challenges due to the design complexity and cost of monitoring events under speculative execution. Mobile and Internet of Things devices have even stricter resource constraints, making adding more hardware performance counters challenging. The trade-off between the number of hardware events and detection accuracy is important to consider. To achieve reasonable accuracy, some works analyze more events than the number of available hardware performance counters, requiring running the application multiple times [31], [36], [37]. This limits the run-time applicability of hardware-based anomaly detection. A careful design of the machine learning classifier is needed to compensate for the reduced characterization due to the fewer events analyzed.

The process of collecting data involves selecting events, extracting features, and reducing them [38]. In the hardware and AI-based anomaly detection context, selecting events consists of choosing from several hardware events available in the processor for use in the anomaly detector. Feature extraction is the process of capturing and storing the registered hardware performance counters (HPCs) in a vector space for analysis by the machine learning (ML) model. Feature reduction, also known as dimensionality reduction, is a form of data processing that deals with redundant dimensions in high-dimensional space. Redundant dimensions contribute to the measurement of noise in the training dataset, which reduces the detection rates of testing.

The selection of events and feature reduction aims to select the most relevant/predictive data for anomaly detection. Since only a few hundred hardware events are available in the processors, it is possible to perform manual event selection through empirical knowledge of each event's representation in the architecture and micro-architecture or based on other studies. However, in some cases, researchers may need this information and start collecting all the hardware events available in the processor, generating big data. In such cases, manual analysis is impossible, and some feature selection techniques must be used. Some commonly used methods include Principal Component Analysis (PCA), Fisher Score [39], Pearson Correlation Coefficient [40] based-techniques, and Information Gain (also called Mutual Information) [41].

A key characteristic of feature extraction is the sampling period of the HPC. There is no set rule for this value. Still, in hardware-based detection experiments, they generally remain in the order of milliseconds or seconds or even as multiples of processor cycles or instruction epochs.

The third fundamental block in the framework is anomaly detection itself, performed by ML classifiers. After training with input data, these classifiers can automatically categorize data into one or more classes (since the training and tested data have similar statistical distributions). Classifiers may differ in terms of the labels available (multi-class and one-class), the way they learn (supervised, unsupervised, and semi-supervised), the mathematical formula/algorithms they implement (Neural Networks-Based, Linear Models-Based, Decision Trees-based, Rule-Based, etc.), and the arrangement of classifiers (multiple stages). As each classifier configuration delivers different results across various metrics (including performance, efficiency, and hardware design overhead), designing an ML classifier is critical.

Multi-class classification-based anomaly detection assumes that the data includes instances labeled as belonging to multiple normal classes during training. When a new instance is tested, it is considered anomalous if it is not classified as normal by any of the trained classifiers. On the other hand, one-class classification-based anomaly detection techniques assume that all the training instances have only one class label, and any new instance that falls outside of the learned boundary is declared anomalous [30].

Supervised learning refers to classifiers trained using labeled examples, while unsupervised learning uses unlabeled examples. Semi-supervised learning combines labeled and unlabeled examples in the dataset [42]. However, obtaining labeled data can be expensive, especially when considering abnormal behaviors, which are often dynamic and challenging to label accurately. Therefore, unsupervised learning classifiers are more commonly used for anomaly detection [30].

Eventually, to improve the accuracy of the classifiers, more complex algorithms such as Ensemble Learning techniques are used [43]. Ensemble Learning is a branch of machine learning that combines the results of multiple base learners to improve decision accuracy. Examples of Ensemble Learning algorithms include Boosting (AdaBoost implementation [44]) and Bagging (Bootstrap Aggregation [45]). Multiple-stage classifiers can also be used to improve the accuracy of anomaly detection [46].

IV. Conclusions

This paper provided an overview of Vitamin-V, a project funded under the Horizon Europe program from 2023 to 2025. The project aims to create a complete open-source virtualization software stack for RISC-V that can be used for cloud services. Among the different activities, one of the main goals of the project is to create a robust virtual execution environment that can be used for software development, validation, verification, and testing thus increasing the trustworthiness of RISC-V cloud applications. Interested readers may follow the project's latest achievements at https://www.vitamin-v.eu.

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. [Online]. Available: https://doi.org/10.1145/3457388.3458657

[4] B. W. Mezger, D. A. Santos *et al.*, "A survey of the RISC-V architecture software support," *IEEE Access*, vol. 10, pp. 51 394–51 411, 2022.

[5] The European Commission. European Processor Initiative. https://www.european-processor-initiative.eu/. Accessed: 2023-03-17.

[6] M. Kovač, "European processor initiative: The industrial cornerstone of eurohpc for exascale era," in *Proceedings of the 16th ACM International Conference on Computing Frontiers*, 2019, p. 319.

[7] Software Freedom Conservancy. RISCV-QEMU. [Online] https://www.qemu.org/. Accessed: 2023-03-17.

[8] "gem5," https://github.com/gem5/gem5, accessed: 2023-03-17.

[9] "Amazon F1," https://aws.amazon.com/ec2/instance-types/f1/, accessed: 2023-03-17.

[10] A. Brokalakis, N. Tampouratzis *et al.*, "Cossim: An open-source integrated solution to address the simulator gap for systems of systems," in *2018 21st Euromicro Conference on Digital System Design (DSD)*, 2018, pp. 115–120.

[11] C. Lattner and V. Adve, "LLVM: A compilation framework for lifelong program analysis and transformation," San Jose, CA, USA, Mar 2004, pp. 75–88.

[12] F. Caforio, P. Iannicelli *et al.*, "Vosysmonitorv: a mixed-criticality solution on linux-capable RISC-V platforms," in *2021 10th Mediterranean Conference on Embedded Computing (MECO)*. IEEE, 2021, pp. 1–4.

[13] N. Tampouratzis and I. Papaefstathiou, "A novel, simulator for heterogeneous cloud systems that incorporate custom hardware accelerators," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 4, pp. 565–576, 2018.

[14] D. Marques, A. Ilic *et al.*, "Application-driven cache-aware roofline model," *Future Generation Computer Systems*, vol. 107, pp. 257–273, 2020.

[15] 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

[16] 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.

[17] J. Lowe-Power, A. M. Ahmad *et al.*, "The gem5 simulator: Version 20.0+," 2020.

[18] "SemiDynamics High Bandwidth Vector-capable RISC-V Cores," https://semidynamics.com/uploads/semidynamics/pdf/2021.03.31-RISCV-WEEK.pdf, accessed: 2023-04-06.

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

[20] 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. [Online]. Available: https://doi.org/10.1145/2046582.2046596

[21] 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.

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

[23] 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.

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

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

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

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

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

[29] 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.

[30] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, jul 2009. [Online]. Available: https://doi.org/10.1145/1541880.1541882

[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. [Online]. Available: https://doi.org/10.1145/2508148.2485970

[32] 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.

[33] 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.

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

[35] 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.

[36] 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. [Online]. Available: https://doi.org/10.1145/3052973.3052999

[37] 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.

[38] C. P. Chenet, A. Savino, and S. D. Carlo, "A survey of hardware-based malware detection approach," 2023.

[39] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*. USA: Wiley-Interscience, 2000.

[40] K. Pearson, "Note on regression and inheritance in the case of two parents," *Proc. R. Soc. Lond.*, vol. 58, pp. 240–242, 1895.

[41] 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.

[42] A. Burkov, *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019. [Online]. Available: https://books.google.it/books?id=0jbxwQEACAAJ

[43] 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.

[44] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002200009791504X

[45] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. -, pp. 123–140, 1996.

[46] 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.