Assessing Runtime Electromagnetic Detection of CPU Hardware Trojans Targeting Kernel Memory

Athanasios Moschos*, Baki Berkay Yilmaz†, Kevin Valakuzhy*, Angelos D. Keromytis*
* Georgia Institute of Technology, School of Electrical and Computer Engineering, USA
† Wayne State University, Department of Electrical and Computer Engineering, USA

***Abstract*—Side-channels are a promising approach for hardware trojan detection, as they can passively reveal malicious hardware activity without requiring additional circuitry or destructive analysis of the device under test. This work investigates the use of electromagnetic (EM) emanations for runtime detection of hardware trojan attacks. Using an open-source hardware trojan that implements an arbitrary memory-write primitive, we tamper with the kernel memory of a Linux system running on a RISC-V microarchitecture and perform a real-time baseline detection using the processor's EM emissions. Our results indicate that, under certain conditions, hardware trojans can be detected indirectly through the anomalous software behavior they enable.**

***Index Terms*—Hardware Trojans, Signal Processing, Operating Systems, Computer Architecture, Electromagnetic Side-channels**

I. Introduction

Detecting hardware trojans during chip deployment is becoming increasingly important due to the absence of industry-wide standards for identifying malicious hardware logic prior to fabrication. This challenge is especially pronounced in CPUs, whose complex operation and diverse microarchitectural activity produce highly variable execution patterns. In this work, we investigate the runtime detection of hardware trojans that target the dynamically allocated kernel memory within CPUs. Kernel memory is particularly important as it hosts security-relevant runtime objects (*e.g.,* task credentials, process descriptors, security policy structures). We show that electromagnetic emissions encode information about the accessed memory region, kernel or user, allowing us to distinguish malicious activity on dynamically allocated kernel memory. Building on this observation, we develop a golden-free runtime detection method that targets such trojan-assisted activity. To the best of our knowledge, this approach to EM-based trojan detection has not been previously explored.

Prior work [1, 2] demonstrated that software-level asymmetries can produce distinct electromagnetic signatures. Specifically, Callan et al. [1] showed how electromagnetic measurements can identify instruction-level execution events. Deviation from expected code-execution patterns is also identifiable [2]. Other EM methods, such as backscattering [3], identify hardware logic modifications directly, but require physical access to the chip surface. In practice, this requirement is often difficult to satisfy for deployed CPUs due to standard packaging and cooling components, such as heatsinks.

II. Tampering With Kernel Memory

Hardware Trojan Design: We use one of the open-source Interrupt-Resilient Trojan designs [4], IRT-1, that features a trigger mechanism implemented within the processor's register file. The trojan's payload circuit is designed to bypass the operating-system-enforced isolation between privileged and non-privileged memory regions. Typically, kernel pages have their user-mode bit cleared to prevent user-level accesses. To enable malicious overwrites from user space, the payload circuit masks the user-mode bit of kernel page table entries (PTEs), thereby suppressing memory-access exceptions.

Attack Emulation: Using a custom Linux kernel module (LKM), we allocate a 4 MB integer array within the kernel space and expose its base address to the user space, similar to a memory-disclosure primitive. Array allocation employs either `kmalloc` or `vmalloc`. A malicious user-space binary can tamper with kernel memory by targeting the disclosed address.

To assess the spectral difference between malicious kernel-memory overwrites and benign user-memory overwrites, the binary allocates a separate 4 MB user-space array through `malloc` and generates random in-bounds indices. At runtime, the binary's overwrite routine targets either the user- or the kernel-space array using those indices. Hence, both scenarios have the *exact same execution flow*. Activating the payload only during the overwrite routine bounds the interval of suppressed memory-access exceptions.

III. EM-Based Detection Methodology

To assess the sensitivity of EM-based monitoring to trojan activity, we adapt the supervised-learning methodology of [5] to a one-class classification approach. Our method comprises three stages. Initially, during a two-stage *training process*, EM activity from various CPU operations is characterized to construct a model of benign EM signatures. In the subsequent *testing stage*, EM signals captured from live CPU operation are processed and compared against this training model in real time to identify anomalous activity.

Training Stage-1: We acquire long EM waveforms, which we partition into shorter time segments, each with a duration equal to the *sample time*. Using a preconfigured *window size* and *overlap ratio*, we compute the Short-Time Fourier Transform (STFT) for each segment. We combine the resulting sequence of time-frequency representations to construct a two-dimensional array analogous to a spectrogram, with one axis being the frequency bins and the other the time.

979-8-3195-1905-4/26/$31.00 © 2026 IEEE

Next, we apply max-pooling to this time-frequency array, motivated by two considerations. First, we expect trojan activity to primarily affect dominant frequencies of the acquired signal. The monitoring system is constantly capturing emanations, unaware of the exact trojan activation interval. Hence, max-pooling preserves the effects of short-lived anomalies appearing in these dominant frequencies. Second, the spectral content at certain frequencies can be unstable, exhibiting smearing across neighboring frequencies. Such behavior could mislead the distance-based classifier of the testing stage, due to the accumulation of small differences across the non-dominant frequency bands. Consequently, max-pooling provides a more compact and robust spectral representation, through a lower-dimensional time-frequency array that preserves anomalous activity in dominant frequencies and reduces sensitivity to local smearing-induced spectral fluctuations.

Another consideration is noise present in the measurement setup. To mitigate noise, we average the rows of the reduced-dimensionality array, producing a single spectral representation. We convert this representation to dB to allow the distance-based classifier to also capture the relative changes occurring in weaker spectral regions. Fluctuations in the global power level across measurements of the same CPU operation can increase variability. We account for this by applying a normalization scheme: we subtract from each single spectral representation its mean and subsequently divide it by its maximum magnitude. Hence, we preserve the overall spectral shape and minimize the effect of global power fluctuations.

Overall, the training stage-1 pipeline is computationally lightweight, and is used both to construct the training-stage EM-signature model and to process the real-time emanations at the testing stage. Consequently, computations in this stage must complete within the waveform acquisition intervals.

Training Stage-2: This stage contributes only to the EM-signature model generation. The stage-1 measurements correspond to distinct CPU operations (*e.g.,* memory accessing), each one with multiple spectral representations. Since normalization rescaled every spectral representation to a comparable range, we apply the k-means algorithm to reduce the size of each EM model associated with a CPU operation. The resulting cluster centroids of each EM model compactly represent the dominant CPU activity patterns, forming a unified EM-signature database for efficient inference during testing.

Testing Stage: To classify whether live EM signals are the outcome of anomalous activity, we compute the ℓ_1-norm distance between each acquired signal (after stage-1 processing) and the database's EM-signatures. The smallest resulting distance is then compared against a predefined threshold, flagging the signal as anomalous if the threshold is exceeded.

IV. Experimental Setup

Trojan Testbed: Similar to [4], we use the 64-bit CVA6 microarchitecture [6] to implement the IRT-1 trojan. The experiments are performed on the Genesys 2 board, with the trojan-afflicted CVA6 operating at a 50 MHz clock and booting a Linux system based on kernel v5.10.7.

TABLE I: Resource utilization of CVA6 and IRT-1.

Module Name	Design LUTs	Design FFs	Design-to-CVA6 LUT Ratio (%)	Design-to-CVA6 FF Ratio (%)
CVA6 w. IRT-1	72,371	46,878	100	100
IRT-1 Trigger	26	1	0.0359	0.0021
IRT-1 Payload	4	3	0.0055	0.0064

The ratio of the malicious logic to the overall size of the trojan-afflicted CVA6 implementation is presented in Table I. Higher ratios yield increased chances of discovering the trojan through its side-channel emanations [7]. In our case, the very small trojan-to-host-design ratio in the generated bitstream challenges our detection approach [8].

Measurement Setup: Our monitoring system uses a Tek-Box 20 mm near-field magnetic probe to measure EM emanations during CVA6's operation. The probe is connected to a B205mini-i SDR for signal acquisition, with an Apple M4 MacBook serving as the real-time processing platform. Similar to [3], we measure frequency ranges around the CVA6 clock harmonics. To determine the optimal probe location and center frequency, we manually search for a location beneath the FPGA with strong EM emissions during the execution of memory operations. We find a favorable center frequency at 265 MHz, near the 5th clock harmonic. The SDR's bandwidth is set to 8 MHz to minimize data throughput and the processing overhead of runtime measurements. Our method is golden-free, hence the trojan-afflicted CVA6 is used to capture measurements during both the training and the testing stage.

V. Results

A. Spectral Representation of Memory Overwrites

To illustrate the spectral characteristics associated with overwrites in the kernel- and user-space memory, we collect EM signals during five distinct scenarios spanning legitimate and malicious activity. We pass the measurements through the training stage-1 processing before plotting their spectrograms.

Kernel Overwrites: We modify the LKM described in Section II to allocate a 4 MB integer array either via `kmalloc` or `vmalloc`, and subsequently perform 300 K back-to-back element overwrites at random indices. The spectral characteristics of this procedure are shown in Fig. 1. Notably, overwrites on the `vmalloc`-allocated object exhibit a significantly stronger EM signature, while taking longer to execute. This temporal and spectral divergence is attributed to differences in the page-table-walking activity.

These two memory-allocation functions rely on inherently distinct virtual-to-physical memory-mapping mechanisms. Regions allocated through `vmalloc` face a substantially higher address-translation overhead, as `vmalloc` constructs a contiguous virtual region of 4 KB physical pages. For a 4 MB-sized array, this results in multiple leaf page table entries. In contrast, `kmalloc` allocates physically contiguous memory from regions covered by the kernel's direct mapping of RAM. Since this mapping is typically backed by huge pages [9], an array of equivalent size is covered by only two 2 MB pages, resulting in more efficient memory accesses.

Fig. 1: Spectrograms of random kernel overwrites in a 4 MB array allocated via `kmalloc` (*top*) and `vmalloc` (*bottom*).

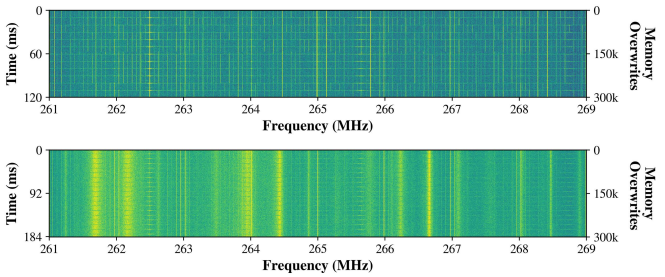
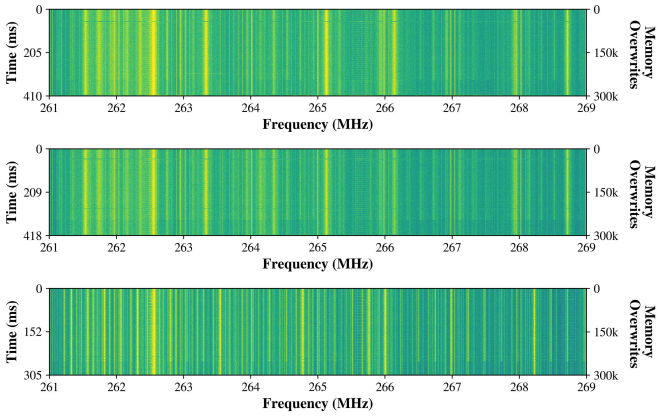


Fig. 2: Spectrograms demonstrating a user process performing benign and trojan-assisted memory overwrites. The *top* shows benign random overwrites on a 4 MB user-space region allocated with `malloc`. The *middle* and *bottom* spectrograms show malicious random overwrites on a 4 MB kernel-space region allocated with `vmalloc` and `kmalloc`, respectively.



User Process Overwrites: For this experiment we use the LKM and the user-space binary described in Section II. We capture measurements while the binary performs benign and malicious overwrites in the allocated arrays. During the overwrite routine, IRT-1 is activated and the exact same code path executes in all cases. The spectrograms for these scenarios can be seen in Fig. 2. The spectral and temporal differences between the top (`malloc`) and middle (`vmalloc`) spectrograms are insignificant, as large user-space allocations obtained through `malloc` are also backed by conventional 4 KB pages. This results in similar demand for address translations and significant page-table-walking activity.

In contrast, the bottom spectrogram (`kmalloc`) presents temporal and spectral discrepancies distinguishable from the other two. This variation is not the outcome of the malicious hardware switching activity, but rather the absence of the page-table-walking activity during the malicious overwrites.

We must note here that pages larger than 4 KB can also be allocated to user-space processes via the kernel feature of transparent huge pages or THP. This could result in fewer address translations, as in the case of `kmalloc`. However, THP support for RISC-V was introduced after Linux v5.14, and is unavailable in our kernel version. Moreover, THP behavior is dependent on the kernel configuration and operating system distribution. Thus, it is outside the scope of our evaluation.

TABLE II: Hyperparameter configurations per training model.

Training Model ID	Size of STFT Window	Overlap Ratio	Max-Pooling Kernel Size	Sample Time (ms)
[1, 2, 3]	1024	0.1	4	[5, 10, 20]
[4, 5, 6]	1024	0.5	4	[5, 10, 20]
[7, 8, 9]	4096	0.1	4	[5, 10, 20]
[10, 11, 12]	4096	0.5	4	[5, 10, 20]
[13, 14, 15]	1024	0.1	32	[5, 10, 20]
[16, 17, 18]	1024	0.5	32	[5, 10, 20]
[19, 20, 21]	4096	0.1	32	[5, 10, 20]
[22, 23, 24]	4096	0.5	32	[5, 10, 20]
[25, 26, 27]	1024	0.1	64	[5, 10, 20]
[28, 29, 30]	1024	0.5	64	[5, 10, 20]
[31, 32, 33]	4096	0.1	64	[5, 10, 20]
[34, 35, 36]	4096	0.5	64	[5, 10, 20]

B. Spectrum-Based Monitoring Evaluation

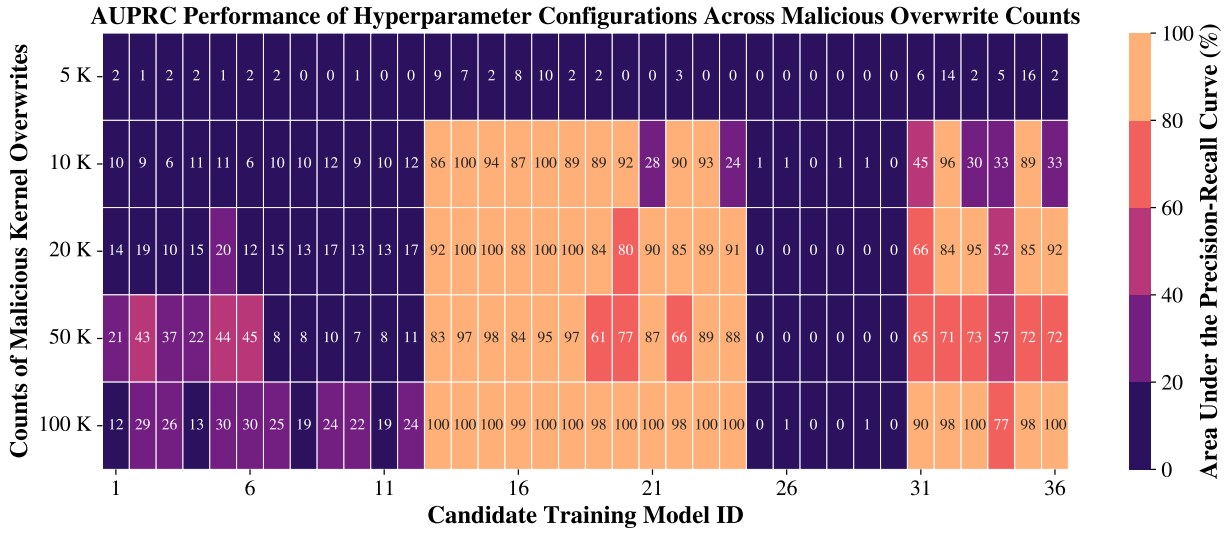
Our training model incorporates the spectral representations shown in Fig. 1 and the user-space overwrite activity of Fig. 2 (top). The malicious activity (middle & bottom) in Fig. 2 is the target behavior that our monitoring system aims to detect during testing and flag it as kernel tampering. The ℓ_1-norm distance metric can reliably distinguish successive overwrites of `kmalloc`-backed kernel objects, owing to their distinct spectral signature. However, the spectral activity generated by successive overwrites of `vmalloc`-allocated objects closely resembles the emissions of benign user-space overwrites, making such tampering events hard to detect with our system.

In practice, the attack duration is not known a priori. A live monitoring system must therefore detect EM anomalies across attacks of varying duration. We approximate this variability by progressively reducing the number of malicious kernel-space overwrites on `kmalloc`-allocated memory and evaluating the resulting detection sensitivity across candidate training models. For this quantitative assessment, we use the area under the precision-recall curve (AUPRC) and summarize the results in the heatmap of Fig. 3. We choose AUPRC due to the dataset imbalance between trojan and benign activity. Table II presents the hyperparameters used to construct the candidate models.

Models 13–24 consistently perform best across malicious overwrite counts from 100 K down to 10 K, with models 14 and 17 achieving 100% AUPRC even at 10 K; only models 21 and 24 show notably degraded performance at this count. These results indicate that the max-pooling kernel size strongly influences detection performance, with the medium-sized value of 32 yielding the best overall results. Models 32 and 35 also achieve comparable AUPRC at 10 K, while using a larger STFT window size of 4096 samples and a large kernel size of 64. Notably, a 10 ms sample time is common to the high-performing models 14, 17, 32 and 35.

Nevertheless, shorter overwrite sequences produce less distinctive spectral patterns, as reflected by the very low AUPRC scores across all models at 5 K overwrites. An attacker who distributes memory modifications across multiple short bursts may therefore evade detection. Future work will focus on improving detection at such low overwrite counts.

Fig. 3: Heatmap evaluating detection capability of distinct training models under varying trojan-assisted memory overwrites.



VI. Discussion

The very small footprint of IRT-1 relative to the overall CVA6, combined with its intertwined placement within the CPU layout, complicates acquisition of a pristine trojan signature at runtime [7, 8]. Spectrally diverse electromagnetic emissions from various microarchitectural components further exacerbate this challenge. Nevertheless, our results indicate that trojan-assisted software execution can produce distinct EM signatures detectable without comparison against trojan-free measurements. We therefore argue that CPU hardware trojans can be detected *through the anomalous software behavior they enable*. This marks a departure from traditional detection approaches that seek to isolate the minute power or EM variations directly attributable to the trojan logic.

To understand whether this shift in detection strategy can practically aid identification of kernel-memory tampering, we examine the prevalence of different kernel memory-allocation mechanisms. We analyzed the most recent Linux kernel source code available at the time of writing (v7.1-rc6) for common allocation APIs, including `kmalloc/vmalloc` and their variants. Our analysis identified approximately 36× more call sites for `kmalloc`-style allocators than for `vmalloc`, with over 36,000 and roughly 1,000 call sites, respectively. These results suggest that untargeted attacks are likely to interact with kernel objects allocated via `kmalloc`-based mechanisms and therefore exhibit detectable emissions. Conversely, carefully scoped modifications of `vmalloc`-backed objects may evade detection, compromising system security. Consequently, favoring `kmalloc`-based allocation for security-sensitive kernel objects could improve the runtime detectability of kernel-memory tampering attacks via EM side-channels.

References

[1] R. Callan, A. Zajic, and M. Prvulovic, "A practical methodology for measuring the side-channel signal available to the attacker for instruction-level events," in *47th Annual IEEE/ACM International Symposium on Microarchitecture*, 2014.

[2] A. Nazari, N. Sehatbakhsh *et al.*, "EDDIE: EM-based detection of deviations in program execution," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 2017.

[3] L. N. Nguyen, B. B. Yilmaz *et al.*, "A novel golden-chip-free clustering technique using backscattering side channel for hardware trojan detection," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2020.

[4] A. Moschos, F. Monrose, and A. D. Keromytis, "Towards practical fabrication stage attacks using interrupt-resilient hardware trojans," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2024.

[5] B. B. Yilmaz, E. M. Ugurlu *et al.*, "Detecting cellphone camera status at distance by exploiting electromagnetic emanations," in *IEEE Military Communications Conference (MILCOM)*, 2019.

[6] F. Zaruba and L. Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2019.

[7] L. N. Nguyen, C.-L. Cheng *et al.*, "Creating a backscattering side channel to enable detection of dormant hardware trojans," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2019.

[8] S. Bhunia, M. S. Hsiao *et al.*, "Hardware trojan attacks: Threat analysis and countermeasures," *Proceedings of the IEEE*, 2014.

[9] J. Corbet. (2022) Solutions for direct-map fragmentation. [Online]. Available: https://lwn.net/Articles/894557/