Assessing Security RISC: Analyzing Flush+Fault Attack on RISC-V Using gem5 Simulator

Mahreen Khan, Maria Mushtaq, Renaud Pacalet and Ludovic Apvrille
Telecom Paris, Institut Polytechnique de Paris, France
fi

Keywords: Microarchitectural Security, Side-Channel Attacks, gem5 Simulator, Embedded Systems, Cache Timing Analysis, Security, Privacy, Complex Systems, RISC-V.

Abstract: Microarchitectural side-channel attacks exploit vulnerabilities such as cache behavior to leak sensitive data. These attacks have been extensively studied on x86 architectures but they remain less explored on RISC-V systems. A recent paper (Gerlach et al., 2023) demonstrated existing and novel microarchitectural attacks on RISC-V hardware platforms (C906, U74, C910, C908). This hardware-based analysis, while realistic, lacks the flexibility and detailed behavioral insights needed to fully understand these attacks. Simulation environments like gem5 (Lowe-Power, 2024) provide fine-grained control and diverse metrics to overcome this limitation and observe the attack in detail. In this paper, gem5 is used to explore Flush+Fault (Gerlach et al., 2023) side-channel attack on RISC-V architecture which was originally tested on RISC-V hardware. Through gem5, we analyze detailed insights of attack such as cache patterns, and timing behaviors. Our results demonstrate the gem5's potential for advancing the understanding of RISC-V microarchitectural vulnerabilities and eventually for developing effective countermeasures.

1 INTRODUCTION

RISC-V architecture is gaining attention for its flexibility and is becoming popular in research and industry. Microarchitectural side-channel attacks exploit small timing differences in hardware to steal sensitive data, such as cryptographic keys. While these attacks have been extensively studied for x86 architectures, there has been less focus on RISC-V systems.

A recent paper (Gerlach et al., 2023) identified several microarchitectural attacks on RISC-V processors like the C906, U74, C910, and C908, including a novel Flush+Fault attack. This attack, a variant of Flush+Reload (Yarom and Falkner, 2014), exploits cache-line faults in RISC-V's split instruction and data cache architecture, revealing instruction cache leakage. This research relied on hardware to highlight RISC-V vulnerabilities but lacked flexibility for deeper insights or countermeasure development. An open-source gem5 simulator can provide fine control over components like caches and branch predictors (Lowe-Power, 2024). Unlike hardware analysis, gem5 enables researchers to test and analyze attack scenarios in controlled, repeatable conditions, offering a better understanding of attack behavior and aiding in the development of countermeasures.

1.1 Contributions

The contributions of this paper are as follows:

- We investigate the return-based Flush+Fault attack on RISC-V architecture using gem5.
- We provide in-depth analysis for Flush+Fault attack behavior.
- We discuss possible countermeasures for the Flush+Fault side-channel attack.
- We demonstrate the importance of gem5 simulations in advancing microarchitectural security.

1.2 Organization

The rest of the paper is organized as follows: Section 2 provides the necessary background on RISC-V side-channel attacks. Section 3 discusses the Flush+Fault side-channel attack mechanism. Section 4 discusses the gem5 simulator. Section 5 presents an overview of related work in the field. In Section 6, we detail our methodology for simulating the Flush+Fault attack within gem5. Section 7 presents our experimental results, followed by a discussion in Section 8. Section 9 outlines potential future research directions. Finally, Section 10 concludes the paper.

Khan, M., Mushtaq, M., Pacalet, R. and Apvrille, L.
Assessing Security RISC: Analyzing Flush+Fault Attack on RISC-V Using gem5 Simulator.
DOI: 10.5220/0013518800003979
In *Proceedings of the 22nd International Conference on Security and Cryptography (SECRYPT 2025)*, pages 607-612
ISBN: 978-989-758-760-3; ISSN: 2184-7711

2 BACKGROUND

This section provides an overview of RISC-V, its cache architecture, and cache side-channel attack principles.

2.1 Overview of RISC-V

RISC-V's modular design enables customization for specific applications. Its minimalistic base instruction set can be expanded with custom extensions. The open-source nature removes licensing fees, making it popular in research, startups, and specialized hardware. Compared to x86 and ARM, RISC-V offers greater flexibility.

2.2 Cache Architectures: x86 vs. RISC-V

Key differences in cache architecture affect security:

- **x86:** L2 and L3 caches often have unified or synchronized I-caches and D-caches (Gerlach et al., 2023). This facilitates attacks like Flush+Reload (Yarom and Falkner, 2014), which exploit shared cache lines to monitor memory access.
- **RISC-V:** Many implementations use a Harvard split-cache architecture (Gerlach et al., 2023), storing instructions and data separately in I-cache and D-cache. This reduces contention and complicates traditional cache attacks.

2.3 Cache Side-Channel Attack Principles

Microarchitectural attacks exploit cache timing variations. In x86, Flush+Reload (Yarom and Falkner, 2014) works by flushing cache lines, monitoring memory accesses, and detecting victim activity based on reload times.

RISC-V's split I-cache and D-cache make traditional Flush+Reload less effective. However, Flush+Fault (Gerlach et al., 2023) exploits instruction cache behavior by using faults to force cache reloads, revealing execution patterns.

These attacks underscore the need for ongoing research into RISC-V microarchitectural security as adoption grows.

3 FLUSH+FAULT ATTACK MECHANISM

Microarchitectural side-channel attacks exploit timing variations to extract sensitive information. The Flush+Reload (Yarom and Falkner, 2014) attack is effective on systems with unified caches but less so on those with separate instruction and data caches like many RISC-V processors (Gerlach et al., 2023). This limitation is addressed by Flush+Fault (Gerlach et al., 2023), a variant that targets instruction cache (I-cache) behavior by leveraging fault-induced timing variations. Instead of reloading, Flush+Fault triggers a fault-triggered jump into the victim's code. This can be done in two ways: (1) a **fault-based** jump that induces an exception in the victim's execution or (2) a **return-based** jump that directly executes a return instruction, avoiding faults.

The attack follows five key steps. First, the attacker flushes the I-cache using the `fence.i` instruction, forcing the CPU to reload instructions from memory. Second, an initial timestamp is recorded. Third, the attacker jumps to an address within the victim's code that may induce a fault, either through an invalid memory address or a faulting instruction. Fourth, once the fault occurs, the attacker intercepts it via a signal handler and records a second timestamp. Finally, by comparing the timestamps, the attacker determines if the victim's code was cached. Faster execution indicates caching, while slower execution suggests a cache miss.

To ensure accurate results, the attacker repeatedly jumps from the same instruction to a dummy address outside the target cache line. This prevents the branch predictor from prefetching the target cache line, which could otherwise eliminate the necessary timing differences and reduce information leakage (Gerlach et al., 2023). Figure 1 summarizes the attack steps.

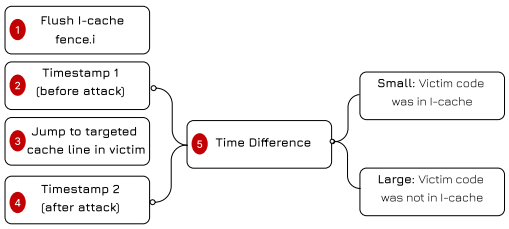

Figure 1: Flush+Fault attack mechanism.

4 GEM5 SIMULATOR OVERVIEW

The gem5 simulator is an open-source tool for microarchitectural and system-level exploration. It provides a flexible environment for modeling and analyzing computing architectures, with configurations managed through Python scripts (Lowe-Power et al., 2020).

4.1 Configuration

gem5 operates in Syscall Emulation (SE) mode and full-system (FS) mode. SE mode is faster and suited for functional validation, while FS mode provides OS-level interactions, essential for security studies (Lowe-Power et al., 2020). gem5 supports multiple CPU models (Lowe-Power, 2024). `AtomicSimpleCPU` is a functional model without cycle accuracy. `TimingSimpleCPU` is an in-order model with instruction timing delays. `O3CPU` is an out-of-order model with speculative execution, making it ideal for security and performance studies (Li et al., 2008). The simulator supports x86, ARM, and RISC-V (Domas, 2017). Users configure ISAs by selecting the target architecture while building gem5. Memory system configuration includes defining cache hierarchies and DRAM models.

4.2 Advantages

gem5's flexibility allows the simulation of multiple ISAs, CPU models, and memory configurations. Its open-source nature enables custom extensions, and its microarchitectural visibility provides fine-grained profiling for execution behavior, cache performance, and branch prediction (Lowe-Power, 2024). This makes it particularly useful for security research, allowing controlled replication of speculative execution and cache-based side-channel attacks.

4.3 Drawbacks

gem5 simulations are significantly slower than real hardware making large-scale studies challenging (Lowe-Power et al., 2020). Multi-core simulations and detailed timing models demand high computational resources. Additionally, while gem5 provides extensive configurability, it may not fully capture real-world timing behaviors, potentially impacting attack feasibility assessments (Qureshi et al., 2021).

Despite these constraints, gem5 remains a valuable tool for studying computing architectures and security vulnerabilities.

5 RELATED WORK

Studying security flaws in microarchitectural components (like caches or branch predictors) using simulation tools such as gem5 is still an emerging field. Early researchers like Low-Power (Lowe-Power et al., 2020; Lowe-Power, 2018) used gem5 to simulate the `Spectre` attack, showing how attackers could trick branch predictors into leaking sensitive data. Later, Pierre Ayoub (Ayoub and Maurice, 2021) improved these simulations for ARM processors, demonstrating how different architectures behave under similar attacks.

While much of this work focuses on `Spectre`, there's little research on simulating newer attacks like `Flush+Fault` (Gerlach et al., 2023) and especially for RISC-V systems. Gerlach et al. (Gerlach et al., 2023) recently discovered the `Flush+Fault` attack and some other novel attacks on real RISC-V hardware, proving that even modern architectures are vulnerable to microarchitectural attacks.

We adapt gem5 to simulate `Flush+Fault` attack on RISC-V, filling a gap in the existing research. By doing this, we aim to get a better understanding of attacks on RISC-V architecture and to be able to develop countermeasures for the newer attacks.

6 SIMULATING FLUSH+FAULT ATTACK ON GEM5

In our analysis using gem5, we will be focusing on the **return-based Flush+Fault** (Gerlach et al., 2023) attack, which leverages the return instruction to minimize overhead and generate cleaner timing signals compared to the traditional fault-based approach.

6.1 Methodology

The methodology for utilizing gem5 to simulate the Flush+Fault side-channel attack is summarized in Figure 2. First, the RISC-V architecture is chosen, and the attack binary for the chosen architecture is prepared. Next, the gem5 simulator is configured. After that, the simulation is run by the gem5 simulator, which generates statistics and results. These results are then analyzed to understand system vulnerabilities and attack behavior (Ta et al., 2018).

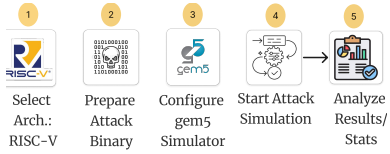

Figure 2: Methodology for Flush+Fault analysis in gem5.

6.2 Preparing Attack Binary

To simulate the Flush+Fault (Gerlach et al., 2023) attack, we first prepared the attack binary using the RISC-V toolchain, cross-compiling the code to target the RISC-V 64-bit ISA. This ensured compatibility with gem5's RISC-V architectural models. The binary was optimized to leverage RISC-V specific instructions and minimize unintended overhead, allowing precise observation of cache and branch predictor behavior during the attack. Next, we configured gem5 to emulate a RISC-V system with parameters tailored to the attack's requirements. The ISA was set to RISC-V with general-purpose and compressed extensions.

6.3 Configuring gem5 Simulator

The CPU model was chosen as the O3CPU (out-of-order execution model) to simulate realistic pipeline dynamics. To analyze the attack's impact on speculative execution, we integrated the LTAGE branch predictor, a high-accuracy predictor that uses global branch history, enabling us to track mispredictions and predictor state changes during the attack sequence. By combining the O3CPU's detailed pipeline simulation, LTAGE predictor metrics, and a realistic cache hierarchy, we could precisely measure how the Flush+Fault attack manipulates cache states and influences branch prediction outcomes. Since the return-based Flush+Fault attack does not require operating system interactions (e.g., kernel-level privileges), we used gem5's System Call Emulation (SE) mode instead of the slower Full System (FS) mode. SE mode provided sufficient isolation for our experiments while significantly accelerating simulation speed. The cache hierarchy we implemented had separate instruction (I) and data (D) caches, each 64 KB in size with 1-way associativity and a 64-byte block size.

6.4 Simulating the Attack

Once the gem5 simulator is properly configured, we simulate the Flush+Fault attack, specifically using the return-based variant. During the simulation, we collect various metrics related to cache performance, branch prediction, and an overview of the simulation execution instructions (Ta et al., 2018). These metrics are recorded in output files and analyzed to detect patterns indicative of side-channel vulnerabilities. Additionally, enabling debugging flags during simulations generates trace files that log detailed events, including memory accesses and instruction executions (Lowe-Power et al., 2020). We capture PipeView O3 traces, which offer detailed information about the pipeline and execution stages. These results are discussed in detail in the following section.

7 RESULTS

This section presents the results of the Flush+Fault (Gerlach et al., 2023) attack on gem5, focusing on key microarchitectural performance metrics. The gem5 simulator was successfully utilized to implement and analyze side-channel attacks, specifically, Flush+Fault (Gerlach et al., 2023) on the RISC-V architecture.

The results obtained align closely with hardware measurements, showcasing gem5's capability to model detailed micro-architectural behavior while providing valuable insights into attack mechanisms. The Flush+Fault (Gerlach et al., 2023) attack in gem5 simulations achieved an F-score of 0.87, demonstrating its accuracy in modeling this attack.

We begin by defining the non-attack scenario, which serves as a baseline distinguisher for comparing attack and non-attack behavior. This provides a reference for identifying deviations introduced by the attack. In the subsequent subsections, we analyze the attack's impact on key architectural components, including cache performance, branch prediction, and instruction execution, with supporting figures to illustrate the observed effects. Finally, we discuss the visualization of the O3CPU pipeline, which enables a detailed examination of execution traces. These traces offer critical insights into the attack's behavior, revealing microarchitectural effects that are difficult to observe through hardware simulations alone.

7.1 No Attack Scenario

To differentiate between attack and non-attack behavior and establish a distinguisher, we implemented a non-attack scenario. In this scenario, only the victim's code executes without any interference from an attacker. The execution follows a normal control flow without intentional cache manipulations, such as explicit instruction cache flushing using `fence.i` in RISC-V. Since no attacker is present, side-channel mechanisms remain inactive, and the system functions as expected, relying solely on standard instruction execution and memory accesses. This baseline serves as a reference point for identifying microarchitectural behavior deviations in attack scenarios.

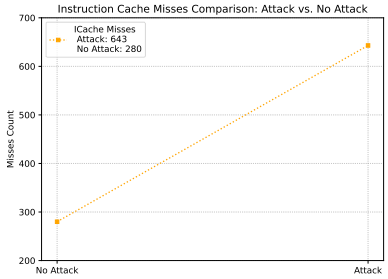

Figure 3: Total ICache misses.

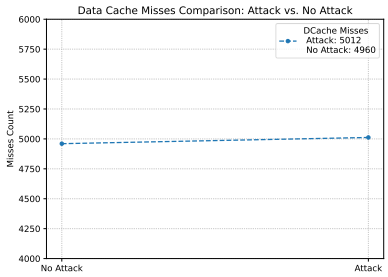

Figure 4: Total DCache misses.

7.2 Cache Performance

Figure 3 shows the impact of the Flush+Fault attack on the Instruction Cache (ICache). During the attack, ICache misses rose from 280 to 643, a significant increase, likely due to `fence.i` instructions flushing the instruction cache and disrupting instruction fetch. On the other hand, DCache misses only increase slightly from 4,960 to 5,012 as shown in Figure 4. This indicates that the attack affects the ICache more than the DCache. These findings demonstrate that such attacks can significantly disrupt instruction fetch operations, highlighting the need for stronger countermeasures, like restricting the use of `fence.i` instructions to privileged access only.

7.3 Branch Predictions

Figure 5 compares the number of branch mispredictions between attack and non-attack scenarios. Under attack conditions, 1,181,977 mispredictions were recorded, compared to 332 in the non-attack scenario. This significant difference highlights attack-induced branch mispredictions, mainly due to the dummy jumps attackers use to avoid speculative prefetching.

7.4 Simulation Execution Metrics

Figure 6 shows the total instruction count executed during attack and non-attack scenarios. The instruction count is significantly higher in the attack scenario (141,078,213 instructions) compared to the non-

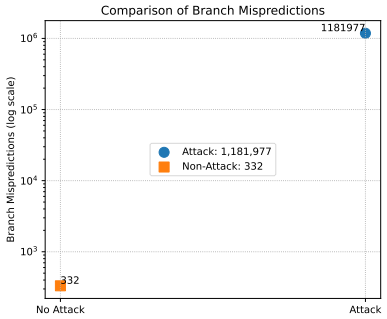

Figure 5: Total branch mispredictions.

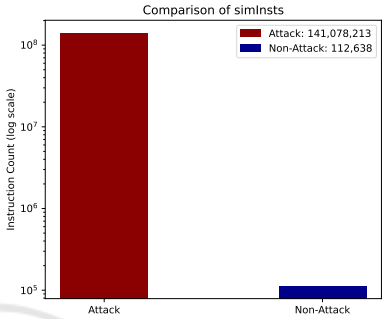

Figure 6: Total Instruction counts.

attack scenario (112,638 instructions). This drastic increase suggests that the attack introduces significant overhead, forcing additional execution cycles and disrupting normal program flow.

7.5 Traces

In gem5, enabling the `O3pipeview` flag captures a complete pipeline execution trace. This generates a detailed sequence of operations as instructions flow through pipeline stages. Tools like `Konata` visualize this trace, as shown in Figure 7.

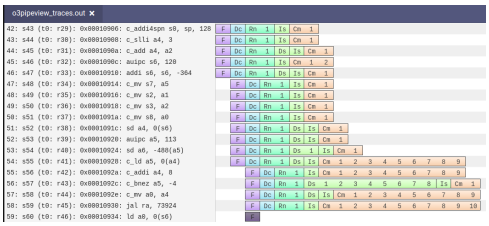

Figure 7: Visualization of the processor pipeline using O3pipeview and Konata.

The pipeline trace proceeds left to right, showing key execution stages: **F** (fetch) retrieves instructions from memory, **Dc** (decode) interprets operations, **Rn** (rename) assigns registers, **Ds** (dispatch) sends instructions to execution units, **IS** (issue) schedules execution, and **Cm** (completion) finalizes execution. This trace helps analyze instruction flow, iden-

tify bottlenecks, and optimize performance.

The analysis highlights how Flush+Fault attacks affect cache timing and branch mispredictions. Using gem5 provides deep insights into attack behavior, aiding in mitigation strategies like strengthening branch predictors and improving cache defenses.

8 DISCUSSION

Our results show that the Flush+Fault attack exposes vulnerabilities in the RISC-V architecture, significantly increasing Instruction Cache (ICache) misses from 280 to 643 (Figure 3). This is likely due to the `fence.i` instruction flushing the cache and disrupting instruction fetch. Data Cache (DCache) misses rise slightly from 4,960 to 5,012 (Figure 4), confirming the attack primarily targets the ICache. Branch prediction is heavily impacted, with the branch mispredictions rising from 332 to 1,181,977 (Figure 5). Our simulation-based approach enables fine-grained analysis, complementing hardware-based research by Gerlach et al. (2023). While gem5 allows controlled studies, simulations may not fully capture real-world timing variations. Future work should validate these effects across different RISC-V implementations.

To mitigate Flush+Fault attacks, restricting access to `fence.i` to privileged users can prevent unauthorized cache flushing. Introducing random execution delays can reduce side-channel exploitability, though at a performance cost. Improving branch prediction algorithms can limit execution flow manipulation. Implementing these countermeasures will enhance RISC-V security against side-channel threats.

9 FUTURE WORK

A key goal is to develop a gem5-based security research platform with flexible cache and pipeline templates. Automated tools could track speculative execution and cache activity, simplifying analysis.

Future work includes adding hardware performance counters in gem5 to monitor events like cache accesses or branch mispredictions. Evaluating security mechanisms such as cache partitioning, robust branch prediction, and instruction randomization could enhance defenses. Studying runtime defenses will help assess performance-security tradeoffs. While gem5 provides cycle-accurate studies close to hardware and detailed microarchitectural insights, future work will test RISC-V side-channel attacks across various simulators and RISC-V hardware to compare results with those from gem5.

10 CONCLUSION

This paper demonstrates that gem5 is a valuable tool for RISC-V security research, enabling controlled analysis of microarchitectural components like caches and branch predictors. By implementing and validating the Flush+Fault attack in gem5, we identified vulnerabilities that could be exploited through side-channel attacks. Our findings show that the attack disrupts instruction caching and branch prediction, increasing cache misses and mispredictions, making timing-based information leakage easier. To mitigate these risks, we suggest restricting access to cache-flushing instructions, improving branch predictor security, and enhancing memory isolation. Future research should focus on real-time attack detection, integrating hardware performance counters in gem5, and exploring secure cache and branch prediction designs. As RISC-V adoption grows, these insights will help shape stronger security measures for modern processors.

REFERENCES

Ayoub, P. and Maurice, C. (2021). Reproducing spectre attack with gem5: How to do it right? In *Proceedings of the 14th European Workshop on Systems Security*.

Domas, C. (2017). Breaking the x86 isa. *Black Hat*, 1:1–6.

Gerlach, L., Weber, D., Zhang, R., and Schwarz, M. (2023). A security risc: microarchitectural attacks on hardware risc-v cpus. In *IEEE S&P*.

Li, J., Tufte, K., Shkapenyuk, V., Papadimos, V., Johnson, T., and Maier, D. (2008). Out-of-order processing: a new architecture for high-performance stream systems. *Proceedings of the VLDB*, 1(1).

Lowe-Power, J. (2018). Visualizing specter with gem5. http://www.lowepower.com/jason/visualizing-spectre-with-gem5.html.

Lowe-Power, J. (2024). Gem5 documentation. https://www.gem5.org/documentation/.

Lowe-Power, J. et al. (2020). The gem5 simulator. *arXiv*.

Qureshi, Y. M., Simon, W. A., Zapater, M., Olcoz, K., and Atienza, D. (2021). Gem5-x: A many-core heterogeneous simulation platform for architectural exploration and optimization. *ACM Transactions on Architecture and Code Optimization (TACO)*, 18(4):1–27.

Ta, T., Cheng, L., and Batten, C. (2018). Simulating multi-core risc-v systems in gem5. In *Workshop on Computer Architecture Research with RISC-V*.

Yarom, Y. and Falkner, K. (2014). {FLUSH+ RELOAD} attack. In *USENIX)*.