

MAD: Microarchitectural Attacks and Defenses

Christopher W. Fletcher^{*1}, Marco Guarnieri^{*2},
David Kohlbrenner^{*3}, and Clémentine Maurice^{*4}

1 University of Illinois – Urbana-Champaign, US. cwfletch@illinois.edu

2 IMDEA Software Institute – Madrid, ES. marco.guarnieri@imdea.org

3 University of Washington – Seattle, US. dkohlbre@cs.washington.edu

4 CNRS – CRISAL, Lille, FR. clementine.maurice@inria.fr

Abstract

Microarchitectural attacks subvert the security assumptions many software-level security mechanisms rely upon, thereby threatening the security of our IT systems. These attacks exploit the side-effects (like subtle timing differences in a program’s execution time) resulting from a processor’s internal optimizations to leak sensitive information and compromise a system’s security. Building systems that are resistant against such attacks requires fundamentally rethinking the design of hardware and software security mechanisms.

This seminar gathered together leading researchers that are working on security at the hardware-software interface spanning four different communities: computer security, computer architectures, programming languages and verification, and applied cryptography. The goals were to (1) present a comprehensive overview of current advances in microarchitectural attacks and defenses, (2) foster interaction and future collaboration between researchers from different research communities, and (3) identify interesting research directions and open challenges that need to be addressed to build the next generation of systems that are resistant to microarchitectural attacks.

Seminar November 26 – December 1, 2023 – <https://www.dagstuhl.de/23481>

2012 ACM Subject Classification Security and privacy → Formal security models; Security and privacy → Security in hardware; Security and privacy → Systems security

Keywords and phrases hardware-software co-design for security, microarchitectural attacks, security architectures, side-channel analysis

Digital Object Identifier 10.4230/DagRep.13.11.151


1 Executive Summary

Christopher W. Fletcher (University of Illinois – Urbana-Champaign, US)

Marco Guarnieri (IMDEA Software Institute – Madrid, ES)

David Kohlbrenner (University of Washington – Seattle, US)

Clémentine Maurice (CNRS CRISAL – Lille, FR)

License  Creative Commons BY 4.0 International license

© Christopher W. Fletcher, Marco Guarnieri, David Kohlbrenner, and Clémentine Maurice

Our society relies on a multitude of information systems that generate, process, and store a massive amount of potentially sensitive data. Protecting and regulating the access to this growing collection of data is critical to prevent security breaches and data misuse. For this, information systems deploy many security mechanisms at different levels: from application-level security checks to, for instance, security mechanisms directly implemented in operating systems. These mechanisms are implemented in a layered fashion where mechanisms at a higher level (say, an application-level security check) rely on the security guarantees provided

* Editor / Organizer



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 4.0 International license

MAD: Microarchitectural Attacks and Defenses, *Dagstuhl Reports*, Vol. 13, Issue 11, pp. 151–166

Editors: Christopher W. Fletcher, Marco Guarnieri, David Kohlbrenner, and Clémentine Maurice



Dagstuhl Reports

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

by lower levels (say, process isolation provided by the operating system). Since the majority of these security mechanisms are implemented in software, their security relies on specific assumptions about how processors execute software.

However, *microarchitectural attacks* have shown, time and again, that many software mechanisms rely on incorrect assumptions about how programs are executed by processors. These attacks, which target the hardware-software interface, exploit the side-effects (like subtle timing differences in a program’s execution time) resulting from a processor’s internal optimizations to compromise a system’s security. Even worse, these attacks clearly highlight that we lack a precise hardware-software interface for security, which is a prerequisite for building trustworthy and reliable security mechanisms.

Scope

The **Dagstuhl Seminar 23481** focused on the topic of **Microarchitectural Attacks and Defenses (MAD for short)**, a rapidly growing research area focused on discovering, mitigating, and preventing microarchitectural attacks. As an indication of this rapid growth, the Spectre [1] and Meltdown [2] papers – two seminal works (published in 2018) illustrating how microarchitectural attacks can bypass and circumvent many software-level security mechanisms – have jointly attracted more than 4500 citations. Since then, researchers from multiple communities – computer security, computer architectures, programming languages and verification, and applied cryptography – have been working on tackling the challenges posed by microarchitectural attacks. In particular, the MAD community has, so far, been broadly focusing on the following research topics:

Attacks: In terms of attack-oriented research, the MAD community has been focusing on characterizing the microarchitectural side-effects arising in modern processors and on identifying new microarchitectural attacks. In particular, the discovery of new microarchitectural details is often the first step towards developing new attacks. Even though the majority of this research still heavily relies on manual analysis and reverse engineering, researchers started to focus also on the development of approaches and tools to automate the discovery of leaks and attacks.

Hardware and software defenses: The MAD community has also been focusing on the development of defenses and mitigations – spanning the entire spectrum from hardware to software – against microarchitectural attacks. For instance, the community has proposed different ways of modifying current microarchitectures to directly prevent microarchitectural leaks, e.g., by identifying (and delaying) those operations that might result in leaks of sensitive information. In terms of software defenses, instead, the community has been focusing on techniques for securely executing computations even on top of current “leaky” processors, e.g., by relying on compiler-based mitigations to prevent leaks.

Foundations and verification: In terms of foundations and verification, the MAD community has been focusing on three core challenges. First, identifying and formalizing new security abstractions capturing microarchitectural leaks. Second, developing automated techniques for reasoning about microarchitectural leaks in software given high-level leakage models. Third, developing verification techniques for proving the security of processors at register-transfer level against microarchitectural attacks.

Goals

The main goal of the **Dagstuhl Seminar 23481 – MAD: Microarchitectural Attacks and Defenses** was to bring together researchers that work on different, but related, research topics such as

1. microarchitectural and side-channel attacks,
2. software security,
3. computer architectures and hardware security,
4. program verification and formal methods for security, and
5. applied cryptography.

For this, the seminar focused on:

1. Providing an overview of the latest research results related with security at the hardware-software interface with a focus on microarchitectural attacks and defenses.
2. Strengthening the interaction between researchers from different community working on topics relevant to microarchitectural attacks and defenses.
3. Discussing relevant open problems about microarchitectural attacks and defenses, identifying novel insights that can arise by combining results from different research areas, and fostering the collaboration between researchers.

Attendees and seminar's structure

The seminar was attended by 35 researchers with diverse background, spanning all research communities related to MAD: computer security, applied cryptography, computer architectures, and programming languages and verification. The attendees were also a good mix between academia (28 attendees) and industry (7 attendees). This mixture of diverse backgrounds, which was particularly appreciated by many participants, led to many interesting discussions fueled by a wide variety of points of views.

The seminar lasted 4.5 days and it was organized as follows. The first two days were dedicated to establishing a common background for all attendees. This was achieved through overview talks on core MAD topics: (a) microarchitectural attacks and defenses, (b) formal methods and verification, (c) defenses at software and hardware level, and (d) a special session dedicated to Rowhammer attacks and defenses. Each overview topic was covered in 2 talks given by leading researchers on the respective topics. The remaining days were dedicated to contributed talks by the attendees (in the mornings) and small discussion groups (in the afternoons). The discussion groups started from topics proposed by the organizers such as “What are the current capabilities of formal methods approaches and which are the challenges for tackling microarchitectural attacks?”, “What is a good methodology for evaluating the security guarantees of microarchitectural defenses?”, or “Which interesting future systems/technologies might have implications for microarchitectural security?”. On the other days, the discussion was directly driven by the attendees, sometimes continuing on the above topics and sometimes exploring other research questions (e.g., identifying a new taxonomy of microarchitectural attacks).

Future plans

Microarchitectural attacks are here to stay: addressing them requires to fundamentally rethink the design of hardware and software security mechanisms. We believe that the core topics of the MAD Dagstuhl Seminar will be relevant and at the edge of research for a

long time. Moreover, the seminar attracted a lot of interest and received positive feedback from the attendees, which particularly appreciated being in contact with leading researchers from other areas working on MAD as well as the presence of both industrial and academic attendees. For these reasons, we believe that this Dagstuhl Seminar should be repeated in the future. Potential improvements for the future editions could be (1) inviting more computer architects and increasing the amount of attendees from industry (in particular, from chip vendors), and (2) dedicating part of the seminar to deep-dives on specific topics.

References

- 1 Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre Attacks: Exploiting Speculative Execution. In *Proceedings of the 40th IEEE Symposium on Security and Privacy (S&P 2019)*.
- 2 Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown: Reading Kernel Memory from User Space. In *Proceedings of the 27th USENIX Security Symposium (USENIX Security 2018)*

2 Table of Contents

Executive Summary

<i>Christopher W. Fletcher, Marco Guarnieri, David Kohlbrenner, and Clémentine Maurice</i>	151
--	-----

Overview of Talks

Microarchitectural defenses in software <i>Sunjay Cauligi</i>	157
Ciphertext Side Channels and their Mitigation <i>Thomas Eisenbarth</i>	157
How can we improve analysis and software mitigation of data-at-rest and value-dependent leakages? <i>Michael Flanders</i>	158
Attacks from Software Leveraging Microarchitectural Features <i>Daniel Gruss</i>	158
Software Defenses: What is the correct interface for a hardware “configuration bit”? <i>David Kohlbrenner</i>	159
What can speculative execution learn from exploitation? <i>Anil Kurmus</i>	159
Modeling and Detecting Microarchitectural Leaks <i>Boris Köpf</i>	159
RowHammer, RowPress and Beyond: Can We Be Free of Bitflips (Soon)? <i>Onur Mutlu</i>	160
Security of PIM (Processing-in-Memory) Systems <i>Onur Mutlu</i>	161
Practical Rowhammer Attacks and Defenses <i>Kaveh Razavi</i>	161
The Gates of Time: Improving Cache Attacks with Transient Execution <i>Eyal Ronen</i>	162
Rowhammer: Learnings from Designing Defenses and Outlook For the Future <i>Gururaj Saileshwar</i>	162
Verified Software Security Down to Gates <i>Caroline Trippel</i>	163
Interrupt-Driven Attacks and Defenses for Microarchitectural Security <i>Jo Van Bulck</i>	163
Hardware attacks and defenses: intro and setting the scene <i>Ingrid Verbauwhede and Jesse De Meulemeester</i>	164

Working groups

Tools for Program Analysis <i>Billy Brumley, Steve Kremer, Moritz Lipp, Nicky Mouha, Alastair Reid, and Jan Reineke</i>	165
--	-----

156 23481 – MAD: Microarchitectural Attacks and Defenses

Open problems

Microarchitectural Side-Channel Mitigations for Serverless Applications	
<i>Aastha Mehta</i>	165
Participants	166

3 Overview of Talks

3.1 Microarchitectural defenses in software

Sunjay Cauligi (MPI-SP – Bochum, DE)

License © Creative Commons BY 4.0 International license
© Sunjay Cauligi

In which I discuss various software-based defenses against Spectre attacks. A successful Spectre exploit is comprised of several distinct phases; different mitigations target these different phases, to varying degrees of completeness and performance. In particular, I highlight the Ultimate SLH [1] and Serberus [2] mitigations and how they are able to overcome the subtleties of transient execution.

References

- 1 Zhang and Barthe and Chuengsatiansup and Schwabe and Yarom. *Ultimate SLH*. USENIX, 2023.
- 2 Mosier and Nemati and Mitchell and Trippel. *Serberus*. Oakland, 2024.

3.2 Ciphertext Side Channels and their Mitigation

Thomas Eisenbarth (Universität Lübeck, DE)

License © Creative Commons BY 4.0 International license
© Thomas Eisenbarth

Joint work of Jan Wichelmann, Anna Pättschke, Luca Wilke, and Thomas Eisenbarth

Main reference Jan Wichelmann, Anna Pättschke, Luca Wilke, Thomas Eisenbarth: “Cipherfix: Mitigating Ciphertext Side-Channel Attacks in Software”, in Proc. of the 32nd USENIX Security Symposium (USENIX Security 23), pp. 6789–6806, USENIX Association, 2023.

URL <https://www.usenix.org/conference/usenixsecurity23/presentation/wichelmann>

In this short talk we discussed memory protection in modern Trusted Execution Environments and the role of logic isolation and/or cryptographic isolation in memory protection. The usage of deterministic encryption enables ciphertext side-channel attacks that can be used to extract secrets from constant-time code. Cipherfix patches binaries by masking all writes of secret values with fresh pseudorandom masks, thereby preventing ciphertext side channels in the protected binary. The induced performance overhead is 2x and more for many workloads. It may serve as a lower bound of the expected costs of moving masking-style countermeasures to arbitrary binaries when trying to prevent arbitrary value leakage on server-grade CPUs, as recently exploited by the Hertzbleed attack.

3.3 How can we improve analysis and software mitigation of data-at-rest and value-dependent leakages?

Michael Flanders (University of Washington – Seattle, US)

License © Creative Commons BY 4.0 International license
© Michael Flanders

Joint work of Michael Flanders, Reshabh Sharma, Alexandra Michael, Dan Grossman, David Kohlbrenner
Main reference Michael Flanders, Reshabh Sharma, Alexandra Michael, Dan Grossman, David Kohlbrenner:
“Avoiding Instruction-Centric Microarchitectural Timing Channels Via Binary-Code Transformations”. ASPLOS 2024, to appear
URL <https://homes.cs.washington.edu/~dkohlbre/papers/cio-asplos24.pdf>

A group of us at UW have been working on detecting and mitigating data-at-rest and value-dependent leakages caused by novel microarchitectural optimizations. These optimizations include things like simplifiable and bypassable computations, silent stores, the Apple data-memory dependent prefetcher, and others as described in the recent “Opening Pandora’s Box ...” paper.

In this talk, I plan to rant about some of the difficulties we faced in implementing leakage analyzers and mitigations in low-level compiler passes and in stand-alone binary analysis tools. These difficulties range from frustrations to soundness issues and arise from improper interfaces and abstractions as well as default assumptions binary analysis tools make that are improper for side-channel analysis. I will briefly discuss our thoughts on solutions but largely want to solicit discussion on better handling of these issues as we see more of these optimizations and accompanying defensive work.

3.4 Attacks from Software Leveraging Microarchitectural Features

Daniel Gruss (TU Graz, AT)

License © Creative Commons BY 4.0 International license
© Daniel Gruss

In this talk, we discuss aspects of attacks from software leveraging microarchitectural features decomposed into multiple parts: We discuss the concept of attacks from software and argue that it ranges from attacks with physical access to attacks where the attacker does not even control a single line of code on the victim system. Thus, threat models for attacks from software vary widely. We discuss that the term microarchitecture also is used in different ways in different contexts: Often it refers specifically to the processor microarchitecture but it is increasingly used as a terminological counterpart to architecture, i.e., microarchitecture as the implementation of an architecture, including anything beneath the architectural interface. Finally, we discuss microarchitectural features that facilitate such attacks and the development trends underlying to the scientific progress in this field.

3.5 Software Defenses: What is the correct interface for a hardware “configuration bit”?

David Kohlbrenner (University of Washington – Seattle, US)

License  Creative Commons BY 4.0 International license
© David Kohlbrenner

With the explosion of novel hardware optimizations, then used for attacks, has come a variety of hardware configuration options for those optimizations. A common approach is a simple on/off bit that can be set in a model specific register (MSR.)

Unfortunately, the preconditions for setting these bits, their effects, and their persistence are decidedly non-uniform. For software-based defenses that intend to use these bits to protect sensitive computation this presents several common problems. Rather than attempt to solve each configuration case on its own, we ask what an ideal simple configuration interface would look like for a compiler-based hardening scheme.

3.6 What can speculative execution learn from exploitation?

Anil Kurmus (IBM Research-Zurich, CH)

License  Creative Commons BY 4.0 International license
© Anil Kurmus

We draw parallels between speculative execution attacks and memory errors. Exploitation of memory errors has a long history, starting from the 1972 Anderson report. While the problem is much more close to being solved in a principled and practical way 50 years later, we have not quite succeeded. What are the lessons we can learn and apply for speculative execution defenses? A few topics of further discussion include “minimum viable patching” vs. principled defenses, attack chaining, attack reliability and portability, taxonomies inspired by memory errors.

3.7 Modeling and Detecting Microarchitectural Leaks

Boris Köpf (Microsoft Research – Cambridge, GB)

License  Creative Commons BY 4.0 International license
© Boris Köpf

Speculative execution attacks such as Spectre and Meltdown exploit microarchitectural optimizations to leak information across security domains. These vulnerabilities often stay undetected for years, because we lack the tools for systematic analysis of CPUs to find them.

In this talk I presented leakage contracts as a way to specify speculative leaks together with Revizor, a tool that can automatically test CPUs against these specifications. I gave examples of how this approach can be used to detect large classes of known and unknown leaks in recent x86 CPUs.

3.8 RowHammer, RowPress and Beyond: Can We Be Free of Bitflips (Soon)?

Onur Mutlu (ETH Zürich, CH)

License © Creative Commons BY 4.0 International license

© Onur Mutlu

Main reference Onur Mutlu, Ataberk Olgun, A. Giray Yağlıkcı: “Fundamentally Understanding and Solving RowHammer”, in Proc. of the 28th Asia and South Pacific Design Automation Conference, ASPDAC '23, ACM, 2023.

URL <http://dx.doi.org/10.1145/3566097.3568350>

We will examine the RowHammer problem in Dynamic Random Access Memory (DRAM), the first example of how a circuit-level failure mechanism can cause a practical and widespread system security vulnerability. RowHammer is the phenomenon that repeatedly accessing a row in a modern DRAM chip predictably causes bitflips in physically-adjacent rows. Building on our initial fundamental work that appeared at ISCA 2014, Google Project Zero demonstrated that this hardware phenomenon can be exploited by user-level programs to gain kernel privileges. Many other works demonstrated other attacks exploiting RowHammer, including remote takeover of a server vulnerable to RowHammer, takeover of a mobile device by a malicious user-level application, and destruction of predictive capabilities of commonly-used deep neural networks.

Unfortunately, the RowHammer problem still plagues cutting-edge DRAM chips, DDR4 and beyond. Based on our recent characterization studies of more than 1500 DRAM chips from six technology generations that appeared at ISCA 2020 and MICRO 2021, we show that RowHammer at the circuit level is getting much worse, newer DRAM chips are much more vulnerable to RowHammer than older ones, and existing mitigation techniques do not work well. We also show that existing proprietary mitigation techniques employed in DDR4 DRAM chips, which are advertised to be Rowhammer-free, can be bypassed via many-sided hammering (also known as TRRespass & Uncovering TRR).

In this talk, we will provide an overview of RowHammer research in academia and industry, with a special focus on recent works that rigorously analyze real chip characteristics and introduce promising solution ideas. We will discuss the effect of RowHammer on High-Bandwidth Memory (HBM) chips and introduce and analyze RowPress, which is a fundamentally different read disturbance phenomenon that also affects all DRAM chips. RowPress greatly (e.g., by 100X) reduces the activation count required to induce bitflips, by keeping an activated row open for a long time. We will also discuss what other problems may be lurking in DRAM and other types of memory, which can potentially threaten the foundations of reliable and secure systems, as memory technologies scale to higher densities. We will conclude by describing and advocating a principled approach to memory robustness (including reliability, security, safety) research that can enable us to better anticipate and prevent such vulnerabilities.

- A short accompanying paper, which appeared at ASP-DAC 2023, can be found here and serves as recommended reading: “Fundamentally Understanding and Solving RowHammer” <https://arxiv.org/abs/2211.07613>.
- Slides: <https://people.inf.ethz.ch/omutlu/pub/onur-DagStuhl-MAD-RowHammer-28-November-2023.pdf>
- A similar talk online on Youtube: <https://www.youtube.com/watch?v=0W7YRRhnunw>

3.9 Security of PIM (Processing-in-Memory) Systems

Onur Mutlu (ETH Zürich, CH)

License © Creative Commons BY 4.0 International license
© Onur Mutlu

Main reference Onur Mutlu, Saugata Ghose, Juan Gómez-Luna, Rachata Ausavarungnirun: “A Modern Primer on Processing in Memory”, CoRR, Vol. abs/2012.03112, 2022.

URL <https://arxiv.org/abs/2012.03112>

PIM systems, which enable various types of computation near (or using) memory structures, are gaining traction. We posit that, on the one hand, different types of PIM systems can cause new security issues, exacerbate known issues, or cause new complications related to security. On the other hand, PIM systems can be used to improve security properties by exposing data less, performing security critical functions in memory, or defining new (and physically smaller) trust boundaries in the system. This talk discusses challenges and opportunities in security of PIM systems.

Some related resources are mentioned below:

- A 2-page overview paper from DAC 2023: “Memory-Centric Computing”, <https://arxiv.org/abs/2305.20000>
- A short vision paper from DATE 2021: “Intelligent Architectures for Intelligent Computing Systems”, <https://arxiv.org/abs/2012.12381>
- A longer survey of modern memory-centric computing ideas and systems (updated August 2022): “A Modern Primer on Processing in Memory”, <https://arxiv.org/abs/2012.03112>
- Slides: <https://people.inf.ethz.ch/omutlu/pub/onur-Dagstuhl-PIM-Security-28-November-2023.pdf>

3.10 Practical Rowhammer Attacks and Defenses

Kaveh Razavi (ETH Zürich, CH)

License © Creative Commons BY 4.0 International license
© Kaveh Razavi

This lecture covers the reverse engineering of in-DRAM Target Row Refresh mechanisms and uses the insights in the development of advanced Rowhammer attacks that bypass these mitigations and the development of principled and scalable alternatives that are secure against these attacks.

3.11 The Gates of Time: Improving Cache Attacks with Transient Execution

Eyal Ronen (Tel Aviv University, IL)

License © Creative Commons BY 4.0 International license
© Eyal Ronen

Joint work of Daniel Katzman, William Kosasih, Chitchanok Chuengsatiansup, Eyal Ronen, Yuval Yarom
Main reference Daniel Katzman, William Kosasih, Chitchanok Chuengsatiansup, Eyal Ronen, Yuval Yarom: “The Gates of Time: Improving Cache Attacks with Transient Execution”, in Proc. of the 32nd USENIX Security Symposium (USENIX Security 23), pp. 1955–1972, USENIX Association, 2023.
URL <https://www.usenix.org/conference/usenixsecurity23/presentation/katzman>

For over two decades, cache attacks have been shown to pose a significant risk to the security of computer systems. In particular, a large number of works show that cache attacks provide a stepping stone for implementing transient-execution attacks. However, much less effort has been expended investigating the reverse direction—how transient execution can be exploited for cache attacks. In this work, we answer this question.

We first show that using transient execution, we can perform arbitrary manipulations of the cache state. Specifically, we design versatile logical gates whose inputs and outputs are the caching state of memory addresses. Our gates are generic enough that we can implement them in WebAssembly. Moreover, the gates work on processors from multiple vendors, including Intel, AMD, Apple, and Samsung. We demonstrate that these gates are Turing complete and allow arbitrary computation on cache states, without exposing the logical values to the architectural state of the program.

We then show two use cases for our gates in cache attacks. The first use case is to amplify the cache state, allowing us to create timing differences of over 100 millisecond between the cases that a specific memory address is cached or not. We show how we can use this capability to build eviction sets in WebAssembly, using only a low-resolution (0.1 millisecond) timer. For the second use case, we present the Prime+Scope attack, a variant of Prime+Probe that decouples the sampling of cache states from the measurement of said state. Prime+Store is the first timing-based cache attack that can sample the cache state at a rate higher than the clock rate. We show how to use Prime+Store to obtain bits from a concurrently executing modular exponentiation, when the only timing signal is at a resolution of 0.1 millisecond.

3.12 Rowhammer: Learnings from Designing Defenses and Outlook For the Future

Gururaj Saileshwar (University of Toronto, CA)

License © Creative Commons BY 4.0 International license
© Gururaj Saileshwar

Rowhammer is a vulnerability affecting newer generations of DRAM (DDR3,DDR4,LPDDR4) where rapid activations of DRAM rows causes bit-flips in neighboring rows. Moreover, recent victim focused mitigation (refreshing victims neighboring aggressor rows) implemented in DDR4 have also been defeated by new attacks.

This talk discusses three recent Rowhammer mitigations proposing new aggressor-focused mitigations – Randomized Row Swap (RRS) [1], Scalable & Secure Row Swap (SRS) [2], and AQUA [3]. Based on learnings from these defenses, this talk summarizes the outlook for Rowhammer mitigations going forward.

References

- 1 Gururaj Saileshwar, Bolin Wang, Moinuddin Qureshi, Prashant J. Nair. *Randomized Row-Swap: Mitigating Row Hammer by Breaking Spatial Correlation between Aggressor and Victim Rows*. ASPLOS 2022. <https://dl.acm.org/doi/10.1145/3503222.3507716>
- 2 Jeonghyun Woo, Gururaj Saileshwar, Prashant J. Nair. *Scalable and Secure Row-Swap: Efficient and Safe Row Hammer Mitigation in Memory Systems*. HPCA 2023. <https://www.computer.org/csdl/proceedings-article/hpca/2023/10070999/1LMbzYX6Uww>
- 3 Anish Saxena, Gururaj Saileshwar, Prashant J. Nair, Moinuddin Qureshi. *AQUA: Scalable Rowhammer Mitigation by Quarantining Aggressor Rows at Runtime*. IEEE MICRO 2022. <https://ieeexplore.ieee.org/document/9923789>

3.13 Verified Software Security Down to Gates

Caroline Trippel (Stanford University, US)

License  Creative Commons BY 4.0 International license
© Caroline Trippel

Hardware-software (HW-SW) contracts are critical for high-assurance computer systems design and an enabler for software design/analysis tools that find and repair hardware-related bugs in programs. E.g., memory consistency models define what values shared memory loads can return in a parallel program. Emerging security contracts define what program data is susceptible to leakage via hardware side-channels and what speculative control- and data-flow is possible at runtime. However, these contracts and the analyses they support are useless if we cannot guarantee microarchitectural compliance, which is a “grand challenge.” Notably, some contracts are still evolving (e.g., security contracts), making hardware compliance a moving target. Even for mature contracts, comprehensively verifying that a complex microarchitecture implements some abstract contract is a time-consuming endeavor involving teams of engineers, which typically requires resorting to incomplete proofs.

Our work takes a radically different approach to the challenge above by synthesizing HW-SW contracts from advanced (i.e., industry-scale/complexity) processor implementations. In this talk, I present our work on: synthesizing security contracts from processor specifications written in Verilog; designing compiler approaches parameterized by these contracts that can find and repair hardware-related vulnerabilities in programs; and updating hardware microarchitectures to support scalable verification and efficient security-hardened programs.

3.14 Interrupt-Driven Attacks and Defenses for Microarchitectural Security

Jo Van Bulck (KU Leuven, BE)


License  Creative Commons BY 4.0 International license
© Jo Van Bulck

Microarchitectural side-channel attacks often face challenges due to limited temporal resolution. Researchers have innovatively employed timer and inter-processor interrupts to temporarily halt victim programs, allowing precise probing of microarchitectural buffers. This technique, while not exclusive to Trusted Execution Environments (TEEs), has demonstrated particular efficacy in such environments.

In this presentation, I share my experiences developing SGX-Step, an open-source framework enabling precise interrupt capabilities within Intel's SGX TEE. I outline specific attack applications of SGX-Step in recent years and its significant impact on the design of effective defenses. Drawing from a thorough root-cause analysis, I explain our collaboration with Intel to devise a hardware-software co-design effectively countering SGX-Step's ability to single-step a victim enclave. Additionally, I highlight our efforts in designing defenses across the system stack for embedded MSP430 Sancus TEE processors. The talk aims to provide insights into interrupt-driven attack evolution and key design choices for mitigating their effects.

3.15 Hardware attacks and defenses: intro and setting the scene

Ingrid Verbauwhede (KU Leuven, BE) and Jesse De Meulemeester (KU Leuven, BE)

License  Creative Commons BY 4.0 International license
© Ingrid Verbauwhede and Jesse De Meulemeester

In this presentation, we introduce hardware, i.e. physical attacks on electronic circuits. With physical security, we mean sensitive information that can be obtained by monitoring or disturbing the physical behavior of the electronic circuit. A first class of attacks are based on passive observation of the data-dependent variations in timing, power consumption or EM emanation. The strength of these attacks is that the device under attack is not aware that it is being observed. A second class of attacks, called fault attacks, actively manipulate the behavior of the integrated circuits. Examples are clock or power glitching, cooling or heating, laser or EM injection, row hammering and more. The effect of these attacks could be transient or permanent. In a second part of the presentation, we give an overview of the effort and lab set-up which is needed to perform these attacks, ranging from simple cheap power probes to laser and FIB set-ups, both for passive and active attacks. In the last part we discussed countermeasures to protect against passive side-channel and active fault attacks against crypto implementations. Countermeasures are split into two main classes. One is hiding, where the goal is to reduce the signal-to-noise ratio of sensitive data. Examples are logic styles as WDDL, clock jitter, instruction shuffling, etc. The second is masking, where sensitive data is randomly split in shares. Operations then work on randomized data and the signal traces do not contain sensitive data that can directly be correlated to the sensitive data. Higher order attacks require higher order masking, i.e. split in a larger number of shares. Countermeasures against fault attacks include on-chip sensors at the circuit level, redundancy and error correcting codes at the algorithm level. Unfortunately, countermeasures against one class of attacks might make the circuit vulnerable to the other class of attacks. Countermeasures resistant to both classes of attacks remain a big research challenge.

4 Working groups

4.1 Tools for Program Analysis

Billy Brumley (Rochester Institute of Technology, US), Steve Kremer (INRIA Nancy – Grand Est, FR), Moritz Lipp (Amazon Web Services – Wien, AT), Nicky Mouha (NIST – Gaithersburg, US), Alastair Reid (Intel – London, GB), and Jan Reineke (Universität des Saarlandes – Saarbrücken, DE)

License © Creative Commons BY 4.0 International license
© Billy Brumley, Steve Kremer, Moritz Lipp, Nicky Mouha, Alastair Reid, and Jan Reineke

A prerequisite to identify microarchitectural attacks and protect against them is to explore various tools that are available to understand program properties. This working group focused on getting some hands-on experience with two specific tools: KLEE and CodeQL.

KLEE is a dynamic symbolic execution engine that can be used to automatically reason about software programs. For example, a programmer can add the `klee_assert(a + b >= a)` statement to determine if there exist values that would cause the addition `a + b` to overflow (thereby making the assertion fail). As an example, KLEE was used to analyze a possible integer overflow in code that was present in OpenSSL’s HKDF implementation.

CodeQL is a static analysis tool that can perform SQL-like queries to look for specific patterns in source code. An application of CodeQL was explored to detect the pattern that caused a buffer overflow vulnerability in an earlier version of the “official” SHA-3 implementation.

5 Open problems

5.1 Microarchitectural Side-Channel Mitigations for Serverless Applications

Aastha Mehta (University of British Columbia – Vancouver, CA)

License © Creative Commons BY 4.0 International license
© Aastha Mehta
Joint work of Yayu Wang, Aastha Mehta

Most of the prior work has focused on microarchitectural side-channel mitigations for cryptographic applications. While cryptography is an important class of applications, we explore microarchitectural side-channel vulnerabilities in other application domains. Specifically, we develop automatic mitigations for serverless applications hosted in cloud platforms. Serverless platforms rely on resource multiplexing among tenants for economies of scale and therefore, coarse-grained resource-partitioning based mitigations are inefficient. Instead, we investigate constant-time execution technique as a principled solution.

Participants

- Gilles Barthe
MPI-SP – Bochum, DE
- Thomas Bourgeat
EPFL – Lausanne, CH
- Billy Brumley
Rochester Institute of
Technology, US
- Sunjay Cauligi
MPI-SP – Bochum, DE
- Chitchanok Chuengsatiansup
The University of Melbourne, AU
- Jesse De Meulemeester
KU Leuven, BE
- Thomas Eisenbarth
Universität Lübeck, DE
- Michael Flanders
University of Washington –
Seattle, US
- Christopher W. Fletcher
University of Illinois –
Urbana-Champaign, US
- Anders Fogh
Intel – Neubiberg, DE
- Daniel Gruss
TU Graz, AT
- Marco Guarnieri
IMDEA Software Institute –
Madrid, ES
- Boris Köpf
Microsoft Research –
Cambridge, GB
- David Kohlbrenner
University of Washington –
Seattle, US
- Steve Kremer
INRIA Nancy – Grand Est, FR
- Anil Kurmus
IBM Research-Zurich, CH
- Moritz Lipp
Amazon Web Services –
Wien, AT
- Aastha Mehta
University of British Columbia –
Vancouver, CA
- Nicky Mouha
NIST – Gaithersburg, US
- Onur Mutlu
ETH Zürich, CH
- Hamed Nemati
CISPA – Saarbrücken, DE
- Yossi Oren
Ben Gurion University –
Beer Sheva, IL
- Riccardo Paccagnella
Carnegie Mellon University –
Pittsburgh, US
- Kaveh Razavi
ETH Zürich, CH
- Alastair Reid
Intel – London, GB
- Jan Reineke
Universität des Saarlandes –
Saarbrücken, DE
- Tamara Rezk
INRIA – Sophia Antipolis, FR
- Eyal Ronen
Tel Aviv University, IL
- Gururaj Saileshwar
University of Toronto, CA
- Michael Schwarz
CISPA – Saarbrücken, DE
- Mark Silberstein
Technion – Haifa, IL
- Caroline Trippel
Stanford University, US
- Jo Van Bulck
KU Leuven, BE
- Ingrid Verbauwhede
KU Leuven, BE
- Hugo Vincent
Arm – Cambridge, GB

