OPINION

# Reflections on Post-Meltdown Trusted Computing

## A Case for Open Security Processors

JAN TOBIAS MÜHLBERG AND JO VAN BULCK

Jan Tobias Mühlberg works as a Research Manager for embedded software security at imec-DistriNet, KU Leuven (BE). His research focuses on protected module architectures such as Sancus, software security, and formal verification and validation of software systems. Tobias is particularly interested in everything safety-critical, IoT security, embedded control systems, and low-level operating system components. He obtained a PhD from the University of York (UK) in 2009. jantobias.muehlberg@cs.kuleuven.be

Jo Van Bulck works as a PhD student at imec-DistriNet, KU Leuven (BE). His research explores hardware-based trusted computing from an integrated attack and defense perspective. He is currently the lead developer of the open-source Sancus architecture, where he is looking into processor design, compiler and operating system infrastructure, and case-study applications. More recently, his focus expanded to investigate architectural limitations and side-channel vulnerabilities in commodity Intel SGX x86 processors. Ultimately, both lines of work come together to establish a hardware-only root-of-trust. jo.vanbulck@cs.kuleuven.be

The recent wave of microarchitectural vulnerabilities in commodity hardware requires us to question our understanding of system security. We deplore that even for processor architectures and research prototypes with an explicit focus on security, open-source designs remain the exception. This article and call for action briefly surveys ongoing community efforts for developing a new generation of *open* security architectures, for which we collectively have a clear understanding of execution semantics and the resulting security implications. We advocate formal approaches to reason about the security guarantees that these architectures can provide, including the absence of microarchitectural bugs and side-channels. We consider such a principled approach essential in an age where society increasingly relies on interconnected and dependable control systems. Finally, we aim to inspire strong industrial and academic collaboration in such an engineering effort, which we believe is too monumental to be suitably addressed by a single enterprise or research community.

The security community has traditionally assessed the trustworthiness of applications at the software level by reasoning about source code as if it were executed on an idealized abstract computing platform. With the advance of hardware-level trusted computing solutions that embed a root-of-trust directly in the hardware, it even becomes possible to abstract away the underlying operating system and supporting software. However, a recent line of microarchitectural attack research, with Rowhammer, Meltdown, and Spectre being prominent examples, revealed fundamental flaws in commodity hardware. These findings range from plain design errors to intricate side-channels and triggered an array of follow-up research, effectively rendering the search for exploitable bugs in commodity processors a playground for researchers who "may have, either directly or indirectly, an economic interest in the performance of the securities of the [affected] companies" (https://amdflaws.com/), and who may or may not act in the public interest with respect to responsible disclosure guidelines. The key lesson to be learned from this wave of microarchitectural vulnerabilities and the tiresome patching process is that current processors exceed our levels of understanding and need to be subjected to independent review and assessment.

Now, having security vulnerabilities in components that are in virtually everyone's computer or phone, and components that are commonly relied upon to build critical infrastructure—think of communications networks, data centers, and cloud systems up to the power grid and hospital equipment—is certainly worrisome. Yet, considering that computing platforms are designed by humans, we have to face that security vulnerabilities are to some extent inevitable. As a community, we must therefore welcome research efforts that enhance our understanding of the attack surface and the limitations of today's commodity computing infrastructure, and that responsibly handle security-related findings to swiftly patch existing systems and avoid introducing similar errors in the future.

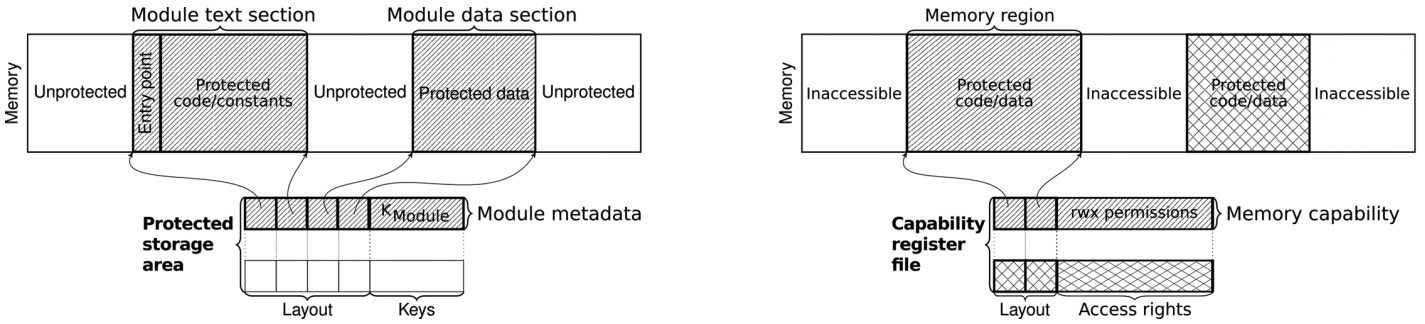


**Figure 1:** Fine-grained intra-address space isolation paradigms. **Left**: Sancus [5] uses the current value of the CPU's program counter to distinguish a protected module (hatched) from untrusted code. The module's *data* memory can only be accessed when executing in the corresponding *text section*, which can only be entered from a single predefined *entry point*. Software attestation is realized through a protected hardware storage area for metadata and cryptographic keys. **Right**: CHERI [6] relies on a dedicated CPU register file for unforgeable *memory capabilities* that provide read/write/execute permissions for individual memory regions (hatched). Flexible application protection domains are defined by deriving more restrictive capabilities at runtime.

## Reverse Engineering Is Insufficient

Conducting this kind of research is far from easy, however, as prevalent business models of the industry hamper such efforts. That is, today's computing platforms are not designed to be analyzed, and intellectual property concerns commonly restrict the freedom of end users (i.e., companies, governments, researchers, the general public) to access hardware design internals, let alone source code. We deplore that even for processor architectures and research prototypes with an explicit focus on security, open-source designs remain the exception [1]. This situation leaves researchers at publicly funded institutions with no choice but to invest enormous reverse-engineering efforts before being able to fully understand the advertised security features, identify limitations and vulnerabilities, or formally prove security properties.

Great examples of such efforts in third-party reverse engineering include the Cambridge formal models [2] of the ARM instruction set architecture, or the fact that the most insightful security analysis of Intel's SGX trusted computing platform comes from MIT researchers [3]. Yet, much of these efforts need to be repeated for every academic publication that models, investigates, or reports on vulnerabilities in closed-source commercial products.

Of course, we acknowledge the importance of intellectual property protection for market shares and revenues in the commercial sector. We also acknowledge the contributions of industry initiatives that integrate strong security features in commodity hardware. Important achievements include secure virtualization extensions, TPM co-processors, and enclaved execution environments such as Intel SGX, ARM TrustZone, and AMD SEV. However, we strongly believe that it is close to impossible for vendors and producers to guarantee the absence of certain classes of critical vulnerabilities in their highly complex products [4].

## Bridging the Trust Gap

We therefore argue that processors in a post-Meltdown world can no longer be considered opaque black boxes that implement an instruction set abstraction. Hardware vendors must not attempt to hide microarchitectural execution semantics but instead allow these details to become part of the specification, so that compilers and operating systems can fully take them into account. When looking at the development of open processors, we welcome a number of such initiatives. For example, a range of free and open-source CPU cores are listed on opencores.org. The RISC-V ISA (https://riscv.org/) enables processor innovation through open standard collaboration, with fully open and industry-competitive RISC-V implementations available.

What we need beyond openness, however, are CPUs with real support for security. We have not fundamentally reconsidered the concepts of hierarchical protection rings and virtual memory since the introduction of the Multics mainframe operating system in 1969. Only very recently have industry and academia developed alternative trusted computing solutions to isolate small software components without relying on privileged system software. As a constructive next step to bridge the trust gap between hardware and software, we envisage enhanced processor designs that allow applications to communicate fine-grained security constraints into the underlying CPU architecture. This would allow microarchitects to apply suitable optimizations while preventing unintended side-channel leakage across protection domains.

Two state-of-the-art secure processor prototypes with an explicit focus on openness are CHERI and Sancus. The CHERI [6] research project explores MIPS extensions for a fine-grained memory capability model. Our own Sancus [5] processor implements open-source (https://distrinet.cs.kuleuven.be/software/sancus/) trusted computing primitives for lightweight embedded applications, such as automotive control systems [7]. Figure

1 compares the CHERI and Sancus approaches to intra-address space isolation. Compared to the legacy Multics virtual memory paradigm, both offer a richer architectural expression of protection domain boundaries. Regarding Spectre- and Meltdown-type speculative execution vulnerabilities, we follow the argument of the CHERI authors [8]. A more explicit architectural notion of protection domains that can be propagated into the microarchitecture has the potential to enable true hardware-software co-design, where the security requirements of the application constrain microarchitectural optimizations.

Importantly, with open security architectures as a prerequisite, dependable hardware-software co-designs can be vetted from a formal perspective. Promising research results include machine-checkable proofs for both functional correctness and high-level integrity and confidentiality security properties [9], or the application of proven-correct analysis to verify the absence of digital side-channels in low-level assembly code. Enhanced hardware description languages such as SecVerilog [10] enable static information flow analysis at hardware design time, which leads to a notion of contractual execution semantics that compilers and applications can rely upon. Using this approach, performant processors can be built, for which the absence of timing side-channels and other undesired information leakage is statically proven. With such trustworthy CPUs as a basis, an especially promising avenue is to apply established techniques in the field of software engineering to develop dependable and highly secure trusted execution environments.

## A Call for Action

Overall, we observe that vulnerabilities in software persist, but the research community has a good understanding of how to address these with established software engineering methods, modern programming languages, and advanced security features in modern processors. However, we also observe that there is a new class of widespread vulnerabilities in commodity hardware ranging from plain design errors to intricate side-channels. These vulnerabilities hamper efforts to improve security on all layers of a system's hardware and software stack. In today's world, where advanced societies increasingly rely on the security and reliability of critical infrastructure in domains such as the power grid, communication, transportation, and medical infrastructure, these vulnerabilities may have disastrous consequences for a great many people, whether exploited through malicious intent or triggered by accident.

We outlined one way to address these threats by relying on open designs and formal methods to develop a new class of secure and dependable processors. As a security community, we will benefit from such an effort by obtaining a shared and clear understanding of the protection mechanisms provided by these processors and of how software systems can be built to make proper use of hardware-level security primitives. It would then become unnecessary for researchers to painstakingly reverse-engineer microarchitectural design details as a prerequisite for exploring new attack techniques or alternative modeling approaches. By reaching the required level of performance while also emphasizing maintainability and rigorous availability guarantees, the envisaged class of processors would form an ideal basis for the design of the networked safety-critical control systems of the future. We believe that architectures such as RISC-V, CHERI, and Sancus present promising starting points for this highly necessary work, and we would like to inspire and invite collaboration in this field.

***Acknowledgments***

This research is partially funded by the Research Fund KU Leuven. Jo Van Bulck is supported by a doctoral grant of the Research Foundation—Flanders (FWO).

***References***

[1] P. Maene, J. Götzfried, R. De Clercq, T. Müller, F. Freiling, and I. Verbauwhede, "Hardware-Based Trusted Computing Architectures for Isolation and Attestation," *IEEE Transactions on Computers*, vol. 67, no. 3 (March 2018), pp. 361–374: https://www.esat.kuleuven.be/cosic/publications/article-2750.pdf.

[2] A. Fox and M. O. Myreen, "A Trustworthy Monadic Formalization of the Armv7 Instruction Set Architecture," in *International Conference on Interactive Theorem Proving* (Springer, 2010), pp. 243–258: https://www.cl.cam.ac.uk/~mom22/itp10-armv7.pdf.

[3] V. Costan and S. Devadas, *Intel SGX Explained* (IACR, 2016), p. 86: https://eprint.iacr.org/2016/086.pdf.

[4] A. Baumann, "Hardware Is the New Software," in *Proceedings of the 16th Workshop on Hot Topics in Operating Systems* (ACM, 2017), pp. 132–137: https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/baumann-hotos17.pdf.

[5] J. Noorman, J. Van Bulck, J. T. Mühlberg, F. Piessens, P. Maene, B. Preneel, I. Verbauwhede, J. Götzfried, T. Müller, and F. Freiling, "Sancus 2.0: A Low-Cost Security Architecture for IoT Devices," *ACM Transactions on Privacy and Security (TOPS)*, vol. 20 (August 2017), pp. 1–33: http://www.beetzsee.de/leuven/2016-acmtops-sancus/paper.pdf.

[6] J. Woodruff, R. N. Watson, D. Chisnall, S. W. Moore, J. Anderson, B. Davis, B. Laurie, P. G. Neumann, R. Norton, and M. Roe, "The CHERI Capability Model: Revisiting RISC in an Age of Risk," in *ACM SIGARCH Computer Architecture News*, vol. 42, no. 3 (June 2014), pp. 457–468: https://www.cl.cam.ac.uk/research/security/ctsrd/pdfs/201406-isca2014-cheri.pdf.

[7] J. Van Bulck, J. T. Mühlberg, and F. Piessens, "VulCAN: Efficient Component Authentication and Software Isolation for Automotive Control Networks," in *Proceedings of the 33rd Annual Computer Security Applications Conference (ACSAC '17),* ACM, 2017, pp. 225–237: https://distrinet.cs.kuleuven.be/software/sancus/publications/acsac17.pdf.

[8] R. N. Watson, J. Woodruff, M. Roe, S. W. Moore, and P. G. Neumann, "Capability Hardware Enhanced RISC Instructions (CHERI): Notes on the Meltdown and Spectre Attacks," University of Cambridge, Computer Laboratory, Technical Report no. 916, 2018: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-916.pdf.

[9] A. Ferraiuolo, A. Baumann, C. Hawblitzel, and B. Parno, "Komodo: Using Verification to Disentangle Secure-Enclave Hardware from Software," in *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17),* ACM, 2017, pp. 287–305: https://www.microsoft.com/en-us/research/wp-content/uploads/2017/10/komodo.pdf.

[10] D. Zhang, Y. Wang, G. E. Suh, and A. C. Myers, "A Hardware Design Language for Timing-Sensitive Information-Flow Security," *ACM SIGPLAN Notices*, vol. 50, no. 4 (May 2015), pp. 503–516: http://www.cse.psu.edu/~dbz5017/pub/asplos15.pdf.