

Inside Risks Through Computer Architecture, Darkly

Total-system hardware and microarchitectural issues are becoming increasingly critical.

SPECTRE,¹¹ MELTDOWN,¹³ FORESHADOW,^{18,21} Rowhammer,⁹ Spoiler,⁹—suddenly it seems as if there is a new and unending stream of vulnerabilities in processors. Previous niche concepts such as speculative execution and cache timing side-channels have taken center stage. Across the whole hardware/software system, new vulnerabilities such as insufficiently protected memory access from untrustworthy PCIe or Thunderbolt USB-C peripherals,¹⁵ malicious Wi-Fi firmware,⁴ or alleged hardware implants¹⁴ are also starting to emerge.

We may be facing a crisis in systems design. What might we do about it? Here, we consider whether existing approaches are adequate, and where substantial new work is needed.

Prove, Don't Patch

Many existing commercial operating systems have extensive vulnerabilities. The MITRE repository of common software security vulnerabilities (CVEs: <http://cve.mitre.org>) currently has over 110,000 open enumerated vulnerabilities that have been reported (excluding ones that have been resolved, and totally ignoring countless other vulnerabilities that have never been reported); the list is growing at a rate of approximately 50 new vulnerabilities each day. Patches cannot possibly keep up with the weaknesses. In addition, patching silicon takes years and potentially costs billions of dol-

lars, which clearly tilts the balance firmly in favor of the attacker.

Recent advances such as the seL4 microkernel,¹⁰ the CertiKOS virtual-machine hierarchy,⁸ and the CompCert verified compiler¹² have significantly contributed to the state of the art in formally proven correctness of operating-system kernels. This technology is not yet widespread, but it offers the potential to prove the absence

of large classes of attacks. It relies on trustworthy models of the architectural abstraction—the hardware/software interface—and those too have advanced recently, in work by the authors and others.^{1,6}

Looking Behind the Hardware Curtain

It has recently become clear that this is not enough, in several ways. First, pro-

cessor hardware (typically subject to extensive verification) has long been assumed to provide a solid foundation for software, but increasingly suffers from its own vulnerabilities. Second, increasing complexity and the way systems are composed of many hardware/software pieces, from many vendors, means one cannot think just in terms of a single-processor architecture. We need to take a holistic view that acknowledges the complexities of this landscape. Third, and most seriously, these new attacks involved phenomena that cut across the traditional architectural abstractions, which have intentionally only described the envelopes of allowed *functional* behavior of hardware implementations, to allow implementation variation in performance. That flexibility has been essential to hardware performance increases—but the attacks involve subtle information flows via performance properties. They expose the hidden consequences of some of the microarchitectural innovations that have given us ever-faster sequential computation in the last decades, as caching and prediction leads to side-channels.

Hardware Vulnerabilities

Ideally, security must be built from the ground up. How can we solve the problem by building the foundations of secure hardware?

For years, *hardware security* to many people has meant focusing on the physical layers. Power/electromagnetic side-channels and fault injection are common techniques for extracting cryptographic secrets by manipulating the physical implementation of a chip. These are not without effectiveness, but it is notable that the new spate of attacks represents entirely different, and more potent, attack vectors.

One lesson from the physical-layer security community is that implementation is critical. Hardware definition languages (HDLs) are compiled down to connections between library logic cells. The logic cells are then placed and routed and the chip layer designs produced. One tiny slip—at any level from architecture to HDL source and compiler, to cell transistor definitions, routing, power, thermals, electromagnetics, dopant concentrations and crystal lattices—can cause a potentially exploitable malfunction. Unlike the binary code of mal-

Designers need to understand more of what takes place in layers above or below their field of expertise.

ware, there is no way to observe many of these physical properties. As a result, systems are more vulnerable to both design mistakes and supply-chain attacks.

As the recent attacks demonstrate, side-channels are becoming more powerful than expected. Traditional physical-layer side-channels are a signals-from-noise problem. If you record enough traces of the power usage, with powerful enough signal processing, you can extract secrets. Architectural side-channels have more bandwidth and better signal-to-noise ratios, leaking much more data more reliably.

If we take a systems-oriented view, what can we say about the problem? First of all, the whole is often worse than the sum of its parts. Systems are composed of disparate components, often sourced from different vendors, and often granting much greater access to resources than needed to fulfill their purpose; this can be a boon for attackers. For example, in Google Project Zero's attack on the Broadcom Wi-Fi chip inside iPhones,⁴ the attackers jumped from bad Wi-Fi packets to installing malicious code on the Wi-Fi chip, and then to compromising iOS on the application processor. Their ability to use the Wi-Fi chip as a springboard multiplied their efficacy. It is surprisingly difficult to reason about the behavior of such compositions of components.⁵ Attackers may create new side-channels through unexpected connections—for example, a memory DIMM that can send network packets via a shared I2C bus with an Ethernet controller.¹⁷

Hardware engineers often talk about 'parasitic' resistance or capacitance—components that were not put there by the designer but were created by the physical implementation, often unhelpfully sucking away signals or power. Today we have parasitic com-

puters. Many components have unintended computational power, which can be perverted—from the x86 page-fault handler² to DMA controllers.¹⁶ This presents a challenge to understanding where all the computation is happening, such as what is software rather than hardware.

Toward Robustly Engineered Trustworthy Systems

Total-system approaches to security defenses are important (see, for example, Bellovin³). A further lesson from physical-layer attacks is why such attacks are not more of a threat today—due to further layers of protection. It is not enough to extract the cryptographic key from a banking card using laser fault injection; the attacker must also use it to steal money. At this point the bank's system-level defenses apply, such as transaction limits and fraud detection. If the key relates only to one account, the payoff involves only money held by that customer, not all other customers. Application-level compartmentalization limits the reward, and thus makes the attack economically nonviable.

Another approach is to ensure that richer contextual information is available that allows the hardware to understand and enforce security properties. The authors are on a team designing, developing, and formally analyzing the CHERI hardware instruction-set architecture,²⁰ as well as CHERI operating system and application security. The CHERI ISA can enable hardware to enforce pointer provenance, arbitrarily fine-grained access controls to virtual memory and to abstract system objects, as well as both coarse- and fine-grained compartmentalization. Together, these can provide enforceable separation and controlled sharing, allowing trustworthy and untrustworthy software (including unmodified legacy code) to coexist securely. Since the hardware has awareness of software constructs such as pointers and compartments, it can protect them, and we can reason about the protection guarantees—for example, formally proving the architectural abstraction enforces specific security properties. We believe this CHERI system architecture has significant potential to provide unprecedented total-system trustworthiness,


including addressing some of the side-channel attacks that were unknown at the time of its conception.¹⁹

Such architectural guarantees enable more secure implementation of currently insecure languages (such as C/C++) and can put demonstrably secure operating-system kernels on a more secure foundation. Similar approaches may apply in other domains, for example between vulnerable components across a system-on-chip.

Engineering such systems requires a more holistic view, with a tighter interplay between hardware, operating systems and applications. In particular, designers need to understand more of what takes place in layers above or below their field of expertise. Better architectural models enable more robust verification of security properties, and amortizing verification costs across projects helps defenders but not attackers. Such verification must be inclusive, testing all the aspects of a system including the boundaries of implementation-defined behavior.

Better verification can defend us against new vulnerabilities present in the abstractions it is based upon, but not against those that involve phenomena that are not modeled. An open question is whether there is an abstraction between an architectural specification and a full hardware implementation that allows us to fully reason about potential leakage, without being so complex as to being intractable.

Conclusion

Traditional models—in which designers have free reign within tightly constrained layers—are no longer fit for purpose. Hardware/software system security architects need better awareness of what comes above and below them, to be able to reason about what happens at other levels of abstraction, and to understand the effects of composition. Managing overall complexity must fully capture information that might be relevant for security analysis, especially for entirely new classes of vulnerabilities. The defensive battle has only just begun. 

References

1. Armstrong, A. et al. ISA Semantics for ARMv8-A, RISC-V, and CHERI-MIPS. In *Proceedings of the Principles of Programming Languages Conference (POPL)* 2019.

2. Bangert, J. et al. The page-fault weird machine: Lessons in instruction-less computation. In *Proceedings of the USENIX Workshop on Offensive Technologies (WOOT)*, 2013.
3. Bellovin, S.M. and Neumann, P.G. The big picture: A systems-oriented view of trustworthiness. *Commun. ACM* 61, 11 (Nov. 2018), 24–26.
4. Beniamini, G. Over The Air: Exploiting Broadcom's Wi-Fi Stack; <https://bit.ly/2oA6GJL>.
5. Gerber, S. et al. Not your parents' physical address space. In *Proceedings of the Hot Topics in Operating Systems Conference (HotOS-XV)* 2015.
6. Goel, S., Hunt, W.A. Jr., and Kaufmann, M. Engineering a formal, executable x86 ISA simulator for software verification. *Provably Correct Systems (ProCoS)*, 2017.
7. Google Project Zero, 2018; <https://bit.ly/2CAQzTMGu>, R. et al. CertiKOS: An Extensible Architecture for Building Certified Concurrent OS Kernels. OSDI 2016, 653–669; See also <https://bit.ly/2Uzj9sI> for ongoing work.
8. Islam, S. et al. SPOILER: Speculative Load Hazards Boost Rowhammer and Cache Attacks, arXiv e-prints (Mar. 1, 2019); <https://bit.ly/2TxWdhk>
9. Klein, G. et al. Comprehensive formal verification of an OS microkernel. *ACM Trans. Computer Systems* 2014; See also <https://bit.ly/2UPKgEY> for ongoing work.
10. Kocher, P. et al. Spectre attacks: Exploiting speculative execution. ArXiv e-prints (Jan. 2018); <https://bit.ly/2lUpJLk>
11. Leroy, X. A formally verified compiler back-end. *Journal of Automated Reasoning* 43, 4 (2009), 363–446.
12. Lipp, M. et al. Meltdown, 2018; <https://bit.ly/2E6myYl>
13. Markettos, A.T. Making sense of the Supermicro motherboard attack; <https://bit.ly/2PqOnld>
14. Markettos, A.T. et al. Thunderclap: Exploring vulnerabilities in operating system IOMMU protection via DMA from untrustworthy peripherals. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS)*, (Feb. 2019).
15. Rushanan, M. and Checkoway, S. Run-DMA. In *Proceedings of the WOOT 2015 Conference*. (2015).
16. Sutherland, G. Secrets of the motherboard ([sh*t] my chipset says). In *Proceedings of the 44CON 2017*, (Sept. 2017).
17. Van Bulck, J. et al. Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution. USENIX Security (Aug. 15–17, 2018); <https://bit.ly/2DusEDT>
18. Watson, R.N.M. et al. Capability Hardware Enhanced RISC Instructions (CHERI): Notes on the Meltdown and Spectre Attacks. Technical Report UCAM-CL-TR-916, University of Cambridge, Computer Laboratory (Feb. 2018); <https://bit.ly/2DuVDrr>
19. Watson, R.N.M. et al. Capability Hardware Enhanced RISC Instructions (CHERI): CHERI Instruction-set Architecture, Version 7, Technical Report UCAM-CL-TR-927, University of Cambridge, Computer Laboratory (Apr. 2019); <https://bit.ly/2XzPgKU>
20. Weiss, O. et al. Foreshadow-NG: Breaking the virtual memory abstraction with transient out-of-order execution (Aug. 2018); <https://bit.ly/2VZLDOh>

A. Theodore Markettos (theo.markettos@cl.cam.ac.uk) is a Senior Research Associate in the Department of Computer Science and Technology at the University of Cambridge, U.K.

Robert N.M. Watson (robert.watson@cl.cam.ac.uk) is a Senior Lecturer in the Department of Computer Science and Technology at the University of Cambridge, U.K.

Simon W. Moore (simon.moore@cl.cam.ac.uk) is Professor of Computer Engineering in the Department of Computer Science and Technology at the University of Cambridge, U.K.

Peter Sewell (Peter.Sewell@cl.cam.ac.uk) is Professor of Computer Science in the Department of Computer Science and Technology at the University of Cambridge, U.K.

Peter G. Neumann (neumann@csl.sri.com) is Chief Scientist of the SRI International Computer Science Lab, and moderator of the ACM Risks Forum.

Copyright held by authors.

Calendar of Events

VOLUTPAT ORNARE ARCU

Donec sit amet neque nec odio pharetra semper. Suspendisse dictum ligula eu diam.

Pellentesque convallis porttitor eros. Nunc placerat accumsan ante. Etiam scelerisque nisl non ligula. Quisque vitae lacus. Pellentesque in augue. Integer laoreet nisl nec ipsum. Ut massa orci molestie quis, blandit et cursus et lorem. Donec congue massa quis metus.

DONEC EU MAGNA

Nunc aliquet ante eget lectus. Vestibulum scelerisque dignissim nisi. Phasellus id elit suspendisse aliquet. Aenean semper, magna quis interdum sagittis, arcu odio tincidunt lacus, non tristique diam arcu sed nibh. Vestibulum non eros vitae dolor dignissim volutpat.

Suspendisse elementum, felis vel hendrerit congue, neque urna consectetur nisl, ac vehicula nisi leo id arcu. Aenean aliquam. Sed suscipit. Quisque semper justo sed leo. Aenean porta, diam non pellentesque pulvinar, ipsum orci ultrices dui, in elementum velit mauris sit amet dolor.

PELLENTESQUE ERAT

Vitae dui semper fermentum. Fusce pede mauris, rutrum at, ullamcorper porta, ultrices ac, felis. Integer nunc enim, bibendum quis, ullamcorper nec, dictum sed, lorem. Morbi lacinia felis vitae massa. Nam tortor magna posuere, adipiscing ac, tincidunt eu, lectus. Nulla tortor nisi, sodales non, luctus non, posuere at, ante. Suspendisse adipiscing sem mollis mi. Duis lobortis commodo orci.

ODIO SED TORTOR

Interdum mollis. Maecenas lobortis, tellus sed mollis nonummy, sapien ante aliquet tellus, et sagittis lacus dolor eu sem. Quisque ut turpis nec risus molestie scelerisque. Nulla placerat. Curabitur sollicitudin quam ut risus.

Mauris aliquet, felis imperdiet adipiscing imperdiet, purus dolor sollicitudin felis, vel convallis ligula lorem scelerisque lorem.