Microarchitecture Security Roundtable

Background, discussion points, and agenda

Table of Contents

Purpose of this Roundtable

This roundtable aims to produce a set of research topics in the area of microarchitecture security. The 2017 Spectre and Meltdown vulnerabilities have shown the world that these problems are real. Given the unique nature of microarchitecture security, topics and research areas should cover many aspects of discovery and mitigation. Through joint research conducted by Red Hat and Boston University, we will determine which technologies can be integrated into upstream projects Red Hat supports, or if necessary into other projects that will provide a long-term home for them.

Schedule

- When: Friday 27 April, 8:30-12:30. Breakfast and lunch will be provided
- Where: Red Hat Boston, room TBD. BlueJeans connection will be provided, but in-person attendance would be preferable if possible.

Attendees

In person:

- David Cantrell (Red Hat)

- Hugh Brock (Red Hat)
- ~~Daniel Gruss~~
- Andrei Lapets (BU)
- Mayank Varia (BU)
- Jon Masters (Red Hat)
- Manual Egele (BU)

Virtual:

- Keith Basil (Red Hat)
- Mike Bursell (Red Hat)
- Daniel Gruss (Graz University of Technology)
- Stefan Mangard (Graz University of Technology)

Sent invitations to following; please move your name to one of above lists (or remove) when you have read this depending on if you can participate or not.

- Dmitri Pal (Red Hat)
- Azer Bestavros (BU)
- Jennifer Stacy
- Jon Masters (Red Hat)
- Peter Jones (Red Hat, software engineer, firmware/early boot)
- Orran Krieger (BU)
- Rich West (BU)
- Ari Trachtenberg (BU)
- W. Clem Karl (BU)

Agenda

- 8:00 - 8:30: Introductions, vision/problem statement from Jon Masters
- 8:30 - 9:35: Research scope discussion through Design Thinking. Focus on research objectives and what projects can be built around these objectives.
- 9:45 - 10:05: Map research areas / projects to upstreams
- 10:05 - 10:45: Detailed technical discussions / next steps game planning
- 11:00 - 12:00: Open time to address parking-lot items from first three hours. Resolve roadmap questions, research areas.
- 12:00- 1:00: Lunch

Background Reading

Please read at least the below before the meeting. Feel free to comment constructively.

Meltdown and Spectre

- https://meltdownattack.com/

ChipLock: support for secure microarchitectures

- https://www.semanticscholar.org/paper/ChipLock%3A-support-for-secure-microarchitectures-Kgil-Falk/1681b7c8c384278870bb2bfcc20c84a540d3bfca

Current Interests

Here’s current thinking on topics:

Chris Wright
Basics of other cache timing side channels associated w/ speculative out of order execution. But also ways of more directly returning sensitive data, or worse, executing controlled content. Flip side might be...interesting work to be done on top of KPTI

Mayank Varia
I am interested in the intersection of cryptography and side channel attacks. More specifically, I have research interests in designing cryptosystems that are resilient to certain types of side channel attacks, such as Meltdown and Spectre. Admittedly (and as a disclaimer) so far in my own research I have focused on power-based side channel attacks against FPGAs/ASICs. From there I have formulated an open question that I think makes sense (but would have a different solution) in the general-purpose CPU setting as well: is it possible to build a large family of ciphers in such a way that each element of the family has independent behavior? That is: even observing how one cipher operates would provide no information on how other ciphers within the family work. If this question applies only to the mathematics of a cipher (i.e., trying to recover the key when observing plaintext-ciphertext pairs), then the construct envisioned by the question is something called a “tweakable cipher.” This object requires its creator to produce two pieces of data: a secret *key* for secrecy along with a public *tweak* that provides (mathematical) variety.
Essentially my question is whether we can form a side-channel-resistant tweakable cipher: a family of ciphers that each have some sort of side channel emanations (e.g. they affect the CPU’s instruction execution or branch prediction decisions), but whose emanations manifest themselves in different ways for each member of the family. The upshot would be that the side channels, while present, would not breach sensitive information such as secret keys or encrypted messages.
This is admittedly a densely- and tersely-written comment, so I’m happy to describe in more detail in person in our roundtable. Also perhaps there are other questions that might be of

interest to me but that I haven't thought about; I'd be happy to hear more about Red Hat's interests.

Manuel Egele
Hypervisor escape vulnerabilities have been a research interest of mine. Inspired by the VENOM (http://venom.crowdstrike.com/, CVE-2015-3456) vulnerability (i.e., memory corruption in QEMU's floppy disk controller), I'd be very interested in exploring an (largely) automated approach that identifies VENOM and more importantly previously unknown (but similar in kind) vulnerabilities in some of the other HW resources that are emulated for today's hypervisors.
My original thoughts for this would build on a dyamic dataflow analysis in QEMU, and importantly covering the emulated HW itself. This would differentiate from existing dataflow systems as these commonly only track dataflows through the guest and do not consider the emulated controller hardware itself.
As the analysis would have to be stimulated with appropriate inputs, this offers a natural synergy with fuzzing, as a way to help identify potential memory corruption vulnerabilities in the emulated HW resources.

Azer Bestavros
I have a very preliminary idea that I would like to discuss/explore more: could software/compiler/formal verification technologies be used to certify that specific vulnerabilities related to side-channel attacks that exploit architectural features be used to either certify that blocks of code are "safe" or to apply transformations to the code that makes it so. For example, if there are known patterns for side channel attacks that exploit a particular feature (say prefetching), could the code be checked for their existence in a block of code, or could one transform the code to break these patterns). Warning: This is fairly sketchy, but that is what roundtables are about :)

Andrei Lapets
Very generally speaking, I am interested in static/compile-time measurement and prediction of costs (including performance costs/overheads) of programs. On the usability side, I am interested in how these can help software engineers and system designers interactively negotiate trade-offs between different dimensions (e.g., performance improvement vs. security guarantees) at design time. It would also be interesting to see how such information can be incorporated into the compilation process. These two are related, as well (e.g., how can programmers specify high-level constraints that will be used to negotiate these trade-offs automatically).

Linda Wang: Please also considered the overhead of these verification methods..
Based on what we learned from SMELT (Spectre & Meltdown), its that the exploit is to take advantage of the prefetch side channel optimization technique/behavior in the modern CPU design.. By fixing it, we ultimately took away the optimization performance gain from implementing the technique. Therefore, any subsequent implementation to protect it via any

security method is going to need to see if it will create less performance degradation compare to the performance gain from the prefetch side channel technique. Otherwise, it is no-op.