Propositions to counter covert channels and speculation vulnerabilities

April 5th, 2022

General considerations

Document purpose

This document is a support for discussions on potential countermeasures against covert channels and speculation vulnerabilities. Nothing is definitive, **everything is open to discussions. Speak up !**

This is a SIG

We must agree on the contours of an extension (or extensions), that's all. The actual work will take place in a TG.

Encumbered IP

We will not discuss competitors' solutions.

Strategy

Independance to microarchitecture specifics

- Instructions do not reference specific microarchitectural mechanisms.
- Make hardware design great again ! Avoid to unnecessarily constrain possible hardware.

Section 2

Threats

Speculation vulnerabilities

Alternative reality

A speculative execution is an execution for an alternative reality inside the same security domain. Speculation must be prevented in critical cases.

The 4 horsemen of the speculation attack

1. Microarchitectural state injection : the attacker prepares the state before speculation.
2. Speculation gadget : gadget triggering speculative execution.
3. Window controlling : the attacker wants to increase the speculation window to reach the disclosure gadget.
4. Disclosure gadget : gadget used to exfiltrate data, with a covert channel or to an “outside” observer.

Section 3

Propositions

Strategy

Security semantics

Add security annotations as an extension, where they are the most relevant. To be used to counter covert channels and speculation attacks, but which may serve other purposes in the future.

The three components of security annotations

1. Explicit security domains.
2. Security properties for memory (coarse).
3. Adversarily-controlled registers.

Details in following slides.

Speculation barrier

An instruction that "stop" speculative execution. Precise definition can be complex, multiple possibilities here. One fantasy example : "speculative execution after the barrier cannot use a speculative value from before the barrier."

Semantic issue

This instruction is dealing with a microarchitectural implementation feature, not a security property.

This instruction is used to reduce speculation windows in critical code segments.

We should not need speculation barriers I

No need for speculation barriers...

It is possible to prevent dangerous speculative execution at the hardware level only, by detecting dangerous patterns [1] or by using dedicated microarchitectural states for speculation [2].

VRoom ! core example

"No speculative fetches to L1/2 caches until they pass VM access".

... but for various reasons, one may want to opt out of these automatic security mechanisms

Most probable reasons : ensuring compatibility with older IPs, prohibitive performance costs of secure execution.

We should not need speculation barriers II

Manually placing security related instructions is bad !

It can only be efficient with perfect knowledge of the hardware. This is a loaded gun given to developers [3]. People will disable it because of the associated performance costs.

Adversary-controlled registers I

We may need to

1. State injection : prevent modifications in uarch state.
2. Speculation gadget : prevent speculation from happening [4].
3. Window controlling : stop ongoing speculation (speculation barrier).
4. Disclosure gadget : prevent usage of uarch state.

reg.ac hint

A `reg.ac rs1` instruction indicates an adversary-controlled register, and hardware must act accordingly. A tentative security-minded generalization of the speculation barrier. `acr` is used to communicate a security property of the application to the hardware.

Adversary-controlled registers II

Adversary-controlled definition

An attacker can bias the register value, not necessarily set it directly.

Bound checks bypass 1

```
# a1 controlled by user,  a0 is array max index
# may prevent speculation on branch
    reg.ac a1
    blt  a0, a1, end
    ...
```

Adversary-controlled registers III

Bound checks bypass 2

```
# a1 controlled by user,  a0 is array max index
    blt  a0, a1, end
    ... # computing address
# t0 contains address to load

# reg.ac may prevent speculative load
    reg.ac t0
    lw   a0, 0(t0)
```

Retpoline example

Retpoline

```
reg.ac a0
jr a0
```

`a0` may not leave traces in the uarch, `jr` may not trigger speculation.

Only the hardware knows how to react in this case, depending on design.
For examples :

- Prevent speculation on `jr`.
- Allow speculation, but only with dedicated uarch structures.
- Prevent speculation plus `jr` execution does not update the uarch structures for jump predictions.
- No-op (simple in-order core).
- ...

Behaviour may be impacted by security policy.

Hardware consequences

Now hardware designers can rely on the following information to create more trustful hardware :

- The privilege levels.
- The address space identifier `ASID`

- ...and its `confidential` bit.
- The security domain and the associated security policy.
- The memory page `confidential` flag.
- The presence of `reg.ac` instructions.

Waiting for your feedback

We need all the feedback we can get :

- Missing features ?
- Useless mechanisms ?
- Opinion on proposed semantics ?
- Remarks on performance impact ?
- ...

Speak up !

References I

[1] Rutvik Choudhary, Jiyong Yu, Christopher Fletcher, and Adam Morrison.
Speculative privacy tracking (spt) : Leaking information from speculative execution without compromising privacy.
In *MICRO-54 : 54th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 607–622, 2021.

[2] Abraham Gonzalez, Ben Korpan, Jerry Zhao, Ed Younis, and K Asanovic.
Replicating and mitigating spectre attacks on an open source risc-v microarchitecture.
In *Third Workshop on Computer Architecture Research with RISC-V (CARRV 2019)*, 2019.

References II

[3] Brian Johannesmeyer, Jakob Koschel, Kaveh Razavi, Herbert Bos, and Cristiano Giuffrida.
Kasper : Scanning for Generalized Transient Execution Gadgets in the Linux Kernel.
In *NDSS*, April 2022.

[4] Allison Randal.
Ghosting the spectre : fine-grained control over speculative execution.
2021.