How Do We Know Weak Memory Matters?

Mike Dodds, *January 2024*

Context: me

2004 → 2017

- York / Cambridge / York - PhD, postdoc, junior prof
- Separation logic, concurrency, weak memory

2017 → now

- Galois Inc - principal scientist
- Parser security, verified cryptography, formal methods at scale

Context: Galois

Transition-focused R&D

- A contract research shop. Paid research by-the-hour
- Mostly security / reliability technologies (PL, formal methods, static analysis)
- Clients: DARPA, US Gov, some commercial

Caricature Galois project

- *Academics write lots of PLDI papers on <THING>*
- *Galois does experiments, builds prototype tools for <THING>*
- *Outcome: evidence of what works in practice wrt <THING>*

“Hello from the peanut gallery…”

- My last weak memory paper: 2019
- I follow the literature (a bit)
- This talk: polemical *(& useful?)*

Mike pitches weak memory research *(a true story)*

Weak memory is very interesting!

- Many systems exhibit weird non-SC behaviors
- Even unsolved how to define the formal *theory*
- Claimed *models* often doesn't match the empirically observed behaviour
- Testing and reasoning *tools* don't take these effects into account

⇒ We should build new *theory*, *models* and *tools*

Mike pitches weak memory research *(a true story)*

Weak memory is very interesting!

- Many systems exhibit weird non-SC behaviors
- Even unsolved how to define the formal *theory*
- Claimed *models* often doesn't match the empirically observed
- Testing and reasoni
effects into account

⇒ We should build nev

"How do we know weak memory matters?"

huh

Galois client context

Clients want *tools* to solve *problems*

They like formal methods, but have limited budgets and lots of problems

They prioritize based on what matters to them

- Not absolute value!
- Based on their current system / threat model / use cases / team …

What matters (for Galois clients)

System-level / user-level impact

The phenomenon generates security / reliability issues pervasively, or for important classes of systems or users

Security relevance

The phenomenon causes serious security problems. Expert teams with good practices suffer from security problems

Inadequate existing practice

Expert teams struggle to build secure / reliable systems. Non-formal approaches do not mitigate the phenomenon, or do so only at great expense

This is a system-centric and security-centric viewpoint

** other viewpoints are also valid*

Many potential FM target domains matter (in this sense)

Eg. memory safety

- *“70% of MS CVEs are memory safety issues”* https://msrc.microsoft.com/blog/2019/07/a-proactive-approach-to-more-secure-code/

Eg. parser security

- “Operation Triangulation” iPhone 0-day (December 2023)
- “Psychic Paper” iMessage 0-day (May 2020)

Eg. micro-architectural timing channels

- MELTDOWN / SPECTRE, and many attacks afterwards

Intuitively, weak memory effects should matter!

- Intimately related to semantics of memory
  → *related to memory safety issues*
- Subtle semantics with many unexpected behaviors
  → *hard for experts to get right*
- Code often runs at high privilege in the kernel
  → *rich opportunities for attacks*
- Models can break language-level abstractions
  → *potential for breaking type-safety / security guarantees*
- Micro-architectural timing attacks have similar causes
  → *'family resemblance' to known-hard security issues*

Examining the hypothesis *“weak memory matters”*

System-level / user-level impact

Do we have examples of system-level reliability / security issues caused by weak memory effects? Are they rare / common? How are they mitigated?

Security relevance

Are there examples of security vulnerabilities or exploits that arise from weak memory? Are they pervasive / important / hard to eliminate?

Inadequate existing practice

Do experts struggle to build reliable / secure systems thanks to weak memory effects? Which experts? What workflows? What cost?

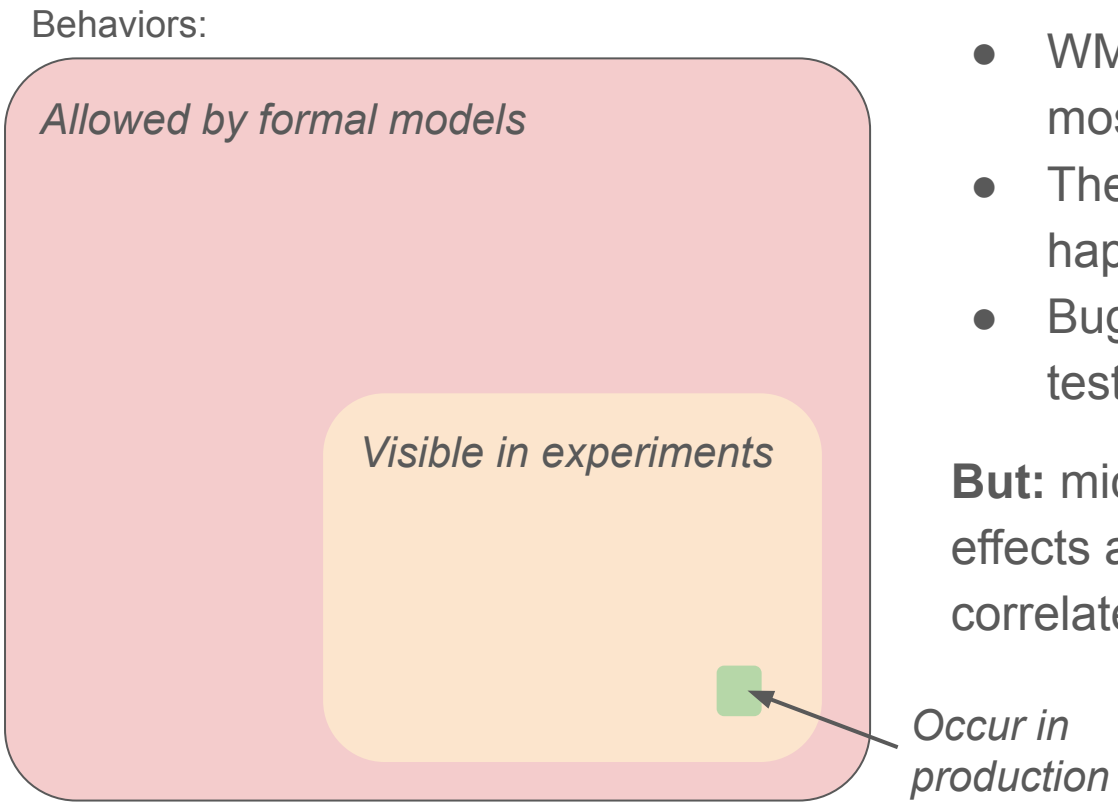
I don’t feel I have compelling evidence

What are some counter-hypotheses?

X-hypothesis #1: *Mike is dumb*

- WM effects matter and there's lots of evidence to support this claim
- Mike just doesn't know about the evidence
- Great!

X-hypothesis #2: *WM effects ~never happen in production*



- WM effects are very rare and mostly uncorrelated
- The weirdest effects only happen in the lab
- Bugs are either caught by testing, or ~ don't happen

But: micro-architectural timing effects are *also* rare and *also* hard to correlate, but they generate attacks

X-hypothesis #3: *WM code is tiny & encapsulated*

Impacted code is mostly core concurrency constructs (locks, schedulers…)

This code is already highly reliable

- Tiny in scope, and heavily audited
- Written by a few super-experts *(hello out there in the audience!)*
- Code often co-designed with weak models (HW, compilers)

This code is deeply encapsulated and therefore difficult to attack

- Concurrency constructs: no untrusted inputs, buried deep in the OS stack
- *vs. crypto libraries: small, highly audited—but takes untrusted inputs*

X-hypothesis #4: *WM developers are very very careful*

WM code is written and audited conservatively, & has too many barriers

&

System-level WM crashes happen, they're just impossible to diagnose as such

But

- How would we establish these things are true? *Could we build tools?*
- Do either of these things mean that WM effects matter?
- Who are these developers? What is the impact at the system / user level?

X-hypothesis #5: *WM bugs are inherently not dangerous*

‘Dangerous’ bugs

- Security failures
- Systematic crashes
- Adversary-controlled memory corruption
- Denial of service

…vs. weak memory bugs

- Very rare mystery crashes?
- Very rare memory corruption?
- Deadlock / fairness issues?

But

- Is this even plausible?
- Is there some deep reason that weak memory bugs are ‘not dangerous’?
- What are typical bug or vulnerability patterns caused by weak memory?

Can we refute these counter-hypotheses?

Are there other explanations?

Wrapping up: *on what matters*

Galois's clients like FM, but they have limited budgets and lots of problems

That means picking problems with *evidence that they matter*

- System-level or user-level impact
- Security relevance
- Inadequate existing practice

I don't feel I have good evidence that WM effects matter in this sense

Clear, compelling evidence would be v interesting and useful

A modest proposal…

Stop building models, start building exploits

Generate evidence WM matters → become vulnerability researchers

Hypothesis: there is an under-explored space of WM security research

- Build system-level CVEs, exploits, case studies that result from weak memory
- Write papers and talks for DEF CON / Black Hat / Oakland / …

Possibly related security topics: secure compilation, ROP / weird machines, micro-architectural timing attacks.

miked@galois.com