Formal methods and security

What to expect when proving?

Security motivated verification.

How do proofs influence security?

Overlay w/ prior paper

Empirical study.

Amazon.

Why proofs for security?

Negative goal: no way to compromise.

Corner cases matter.

Proofs: consider all cases.

Machine-checked.

Code irrelevant; consider spec.
Why not proofs for security?

- Lots of effort ⇒ alternatives?
  - Fuzzing
  - Audits
  - Refactor/Isolation

- Idiosyncratic code.
  - Less clean.
  - Error-prone in ways that bypass the spec.

Threat model.

- Pedantic + incorrect threat model.

Wrong assumptions.
- Imprecise CPU model.
  - Missing timing.
  - Missing non-det.
  - Missing features.

- Physical attacks ⇒ Rowhammer

Attackers bypass known defenses.
- Phishing.
- Side channel.
- Debug/mgmt interfaces.
- Physical attacks.

- Hard to interpret theorem.
  - bcpierce: 1 week to read the self-tau.
<table>
<thead>
<tr>
<th>Code changes</th>
<th>To verify?</th>
<th>Unreadable code?</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Yes (P)</td>
<td>P</td>
</tr>
<tr>
<td>Yes (A)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To make the sys. secure:

- Misguided theorem?
- Overhead of proofs (broke security?)
- Great!

Breakout Q: experience / verify kube?"
Qualified guarantees
Forces precise threat model.
Conditional correctness/security.
→ Hard to extract assumptions.
→ Hard to understand implications.

ARM ISA.
(Over-)simplified: UNSPEC.

Structured exploration.
Forces dev. to think precisely.
Butler’s steps:
1. State
2. Spec
3. Abstraction
4. Proof

Facebook Infer.
IronFleet
Everest

Amazon.
NetCore
I4.
Side effects

- Significant effort?
- Awkward code?
- Awkward shim layer?

Over-reliance on proven defenses

- Java VM / Javascript engine
  - Isolation: mem. safety.
  - Measures: ASLR, canaries, overflow checks, ...

"Defense in depth":
- Mitigation of wrong threat model.