WiP: Automatic Transient Execution Attack Detection

Zack McKevitt
zachary.mckevitt@colorado.edu

Tamara Lehman
tamara.lehman@colorado.edu

Ashutosh Trivedi
ashutosh.trivedi@colorado.edu

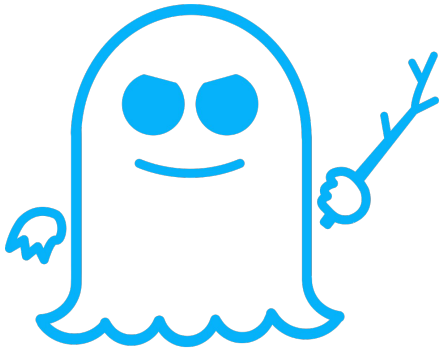
University of Colorado Boulder
Electrical, Computer, and Energy Engineering

Spectre Attacks

- Processor level exploit that relies on **speculative execution**
- Speculate some secret value and observe it before verifying permission
- Unmitigated at hardware level

```
1 void victim_function(size_t x) {
2     if(x < array1_size) {
3         temp &= array2[array1[x] * 512];
4     }
5 }
```

Key Contributions



- Create platform for detecting spectre vulnerabilities and notifying software developers of risky behavior
- Analyze software and microarchitecture (hardware) in combination
- Synthesize a near-regular formal language that explains existing vulnerabilities and learns new ones

Key Contributions

- Create platform for detecting spectre vulnerabilities and notifying software developers of risky behavior
- Analyze software and microarchitecture (hardware) in combination
- Synthesize a near-regular formal language that explains existing vulnerabilities and learns new ones

Motivation

- Prior work analyzes primarily microarchitecture (SPEECHMINER [1]), or software (SPECTECTOR [2]).
- Our work focuses on probabilistic programming to **learn** spectre vulnerabilities
- Learning based approach vs experiment based approach
- Goal: does a given code sample potentially leak information via spectre based attacks?

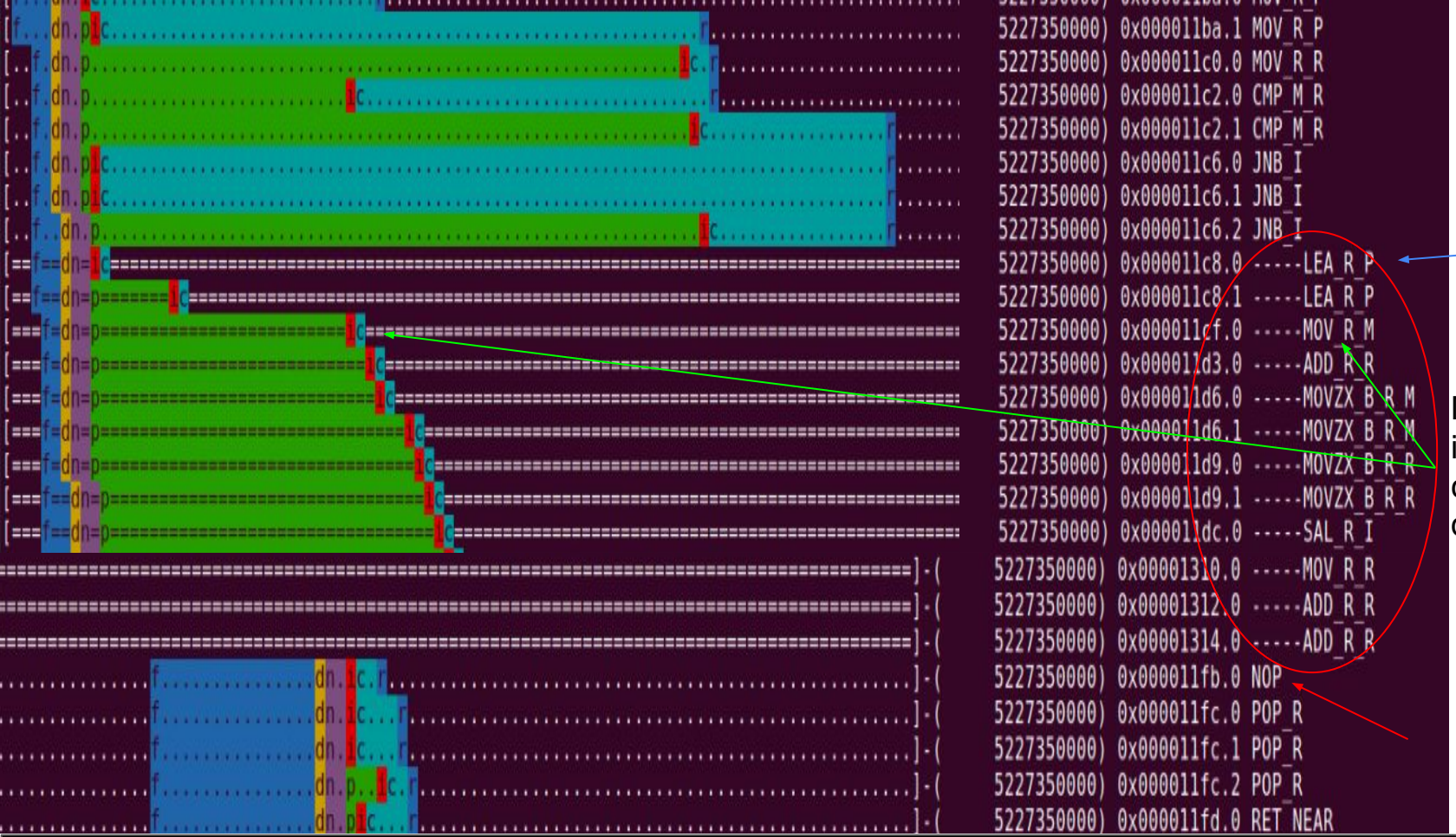
[1] Xiao, Y., Zhang, Y., & Teodorescu, R. (2019). SPEECHMINER: A Framework for Investigating and Measuring Speculative Execution Vulnerabilities.

[2] Guarnieri, M., Köpf, B., Morales, J. F., Reineke, J., & Sánchez, A. (2019). SPECTECTOR: Principled Detection of Speculative Information Flows.

Methodology

- Use the gem5 simulator [3] to simulate code on an x86 out of order processor
- Traces allow us to analyze microarchitectural state
- Use spectre safe and spectre vulnerable programs as training data
- Use probabilistic programming languages (PPLs) to learn an automaton that flags vulnerable programs

[3] Lowe-Power, J., Ahmad, A. M., Akram, A., Alian, M., Amslinger, R., Andreozzi, M., Armejach, A., Asmussen, N., Beckmann, B., Bharadwaj, S., Black, G., Bloom, G., Bruce, B. R., Carvalho, D. R., Castrillon, J., Chen, L., Derumigny, N., Diestelhorst, S., Elsasser, W., … Zulian, É. F. (2020). The gem5 Simulator: Version 20.0+.



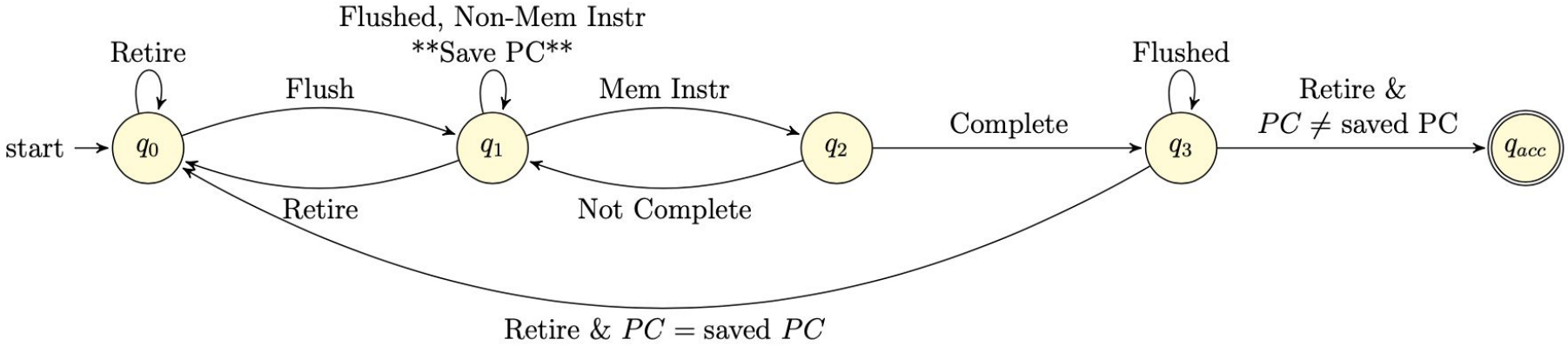
First flushed instruction, save PC

Memory instruction that completes but doesn't retire

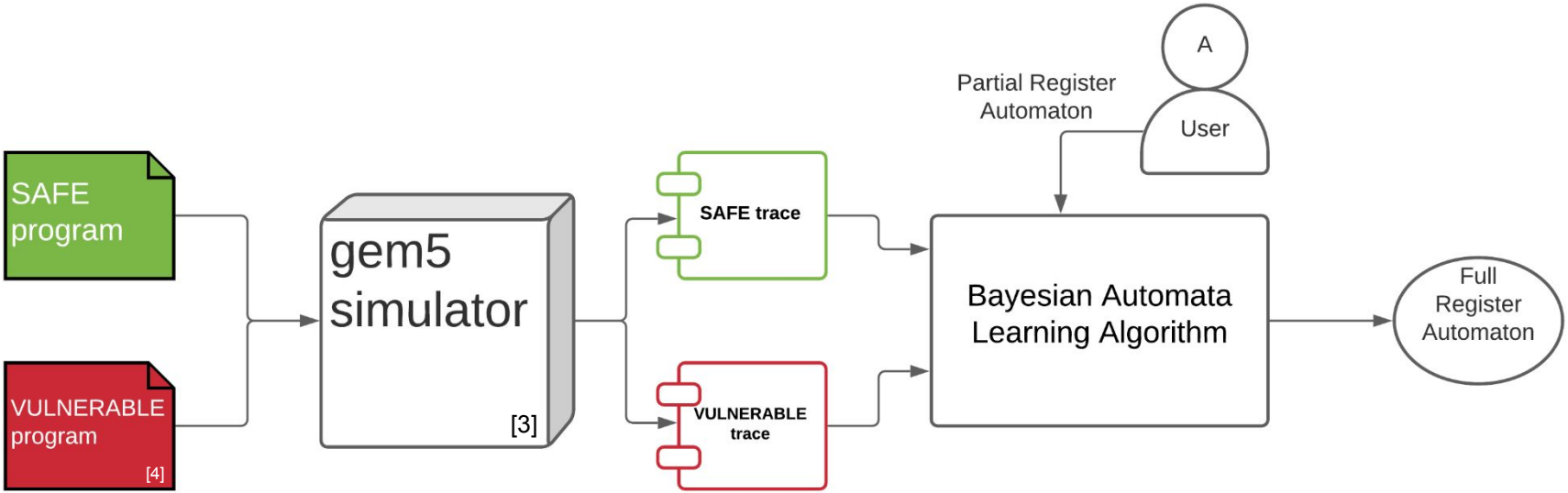
Resume execution at different PC

[4] Jason Lowe-Power. 2018. Visualizing Spectre with gem5. Blog post: http://www.lowepower.com/jason/visualizing-spectre-with-gem5.html.

Automata Learning



- Using PPLs to learn a single store register automaton that recognizes desired language
- Training data is made up of safe and vulnerable traces used as examples/counterexamples for our language



[3] Lowe-Power, J., Ahmad, A. M., Akram, A., Alian, M., Amslinger, R., Andreozzi, M., Armejach, A., Asmussen, N., Beckmann, B., Bharadwaj, S., Black, G., Bloom, G., Bruce, B. R., Carvalho, D. R., Castrillon, J., Chen, L., Derumigny, N., Diestelhorst, S., Elsasser, W., … Zulian, É. F. (2020). The gem5 Simulator: Version 20.0+.

[4] Jason Lowe-Power. 2018. Visualizing Spectre with gem5. Blog post: http://www.lowepower.com/jason/visualizing-spectre-with-gem5.html.

Future Plans

- Develop working model for automaton learning in Pyro
- Generate sample of traces of multiple spectre variants
- Convert gem5 traces into string stream and classify training set
- Learn an automaton from training data

Q&A