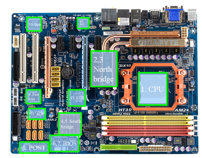
Tamper Evident Microprocessors

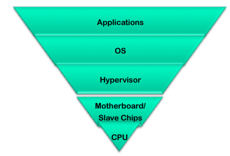
Adam Waksman
Simha Sethumadhavan

Computer Architecture & Security Technologies Lab (CASTL)
Department of Computer Science
Columbia University

Modern Hardware is Complex

- Modern systems built on layers of hardware

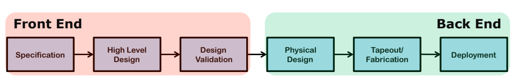




- Complexity increases risk of backdoors
  - More hands
  - Easier to hide
- A significant vulnerability
  - Hardware is the root of trust
  - All hardware and software controlled by microprocessors

Prior Work and Scope

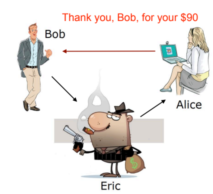
- Microprocessor design stages



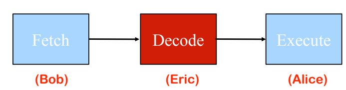
- Prior work focuses on back end
  - More immediate threat
  - Example: IC fingerprinting [Agrawal et al., 2007]
- Front end is the extreme root
  - Common assumption: golden model from front end
  - Focus of this work

Key Idea: Use Inherent Division of Work

- Bob
  - Nice Guy
  - Donates $100
- Eric
  - Evil Accountant
  - Steals $10
- Alice
  - Charity President
  - Receives $90



Microprocessor Pipeline Stages Analogue

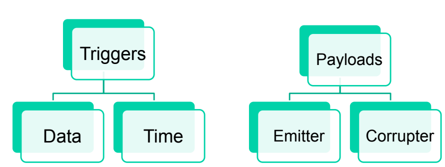


Outline

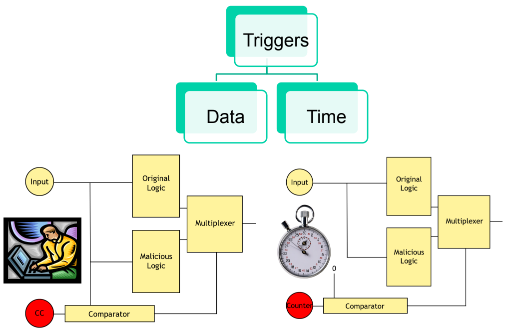
- Taxonomy
  - Ticking Timebombs, Cheat Codes, Emitters, Corrupters
- Solutions
  - TrustNet and DataWatch
- Results
  - Correctness, Coverage and Costs
- Future Work

Taxonomy of Attacks

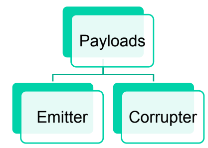
- Backdoor = Trigger + Payload
  - Trigger: Turns on an attack
  - Payload: Malicious, illegal action



Taxonomy of Attacks: Triggers

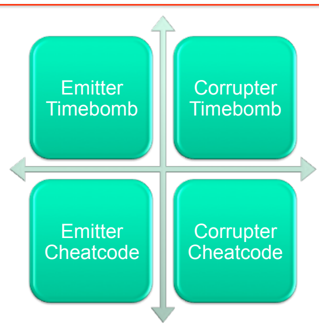


Taxonomy of Attacks: Payloads



- Emitter Attacks
  - Extra malicious events
  - Separate from normal events
- Corrupter Attacks
  - No extra malicious events
  - Normal instructions altered

Taxonomy of Attacks: Summary



Assumptions

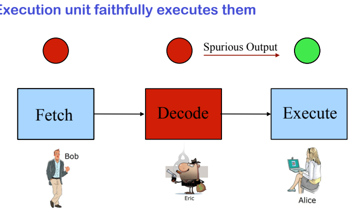
- Large design team
  - Each designer works on one unit or part of one
  - Security add-ons cannot be done by one member
- Full knowledge
  - Attacker has complete access to all design specifications
  - Attacker also knows about additional security mechanism
- Equal distrust
  - Any one designer/unit may be evil
  - Security add-ons may contain backdoors

Outline

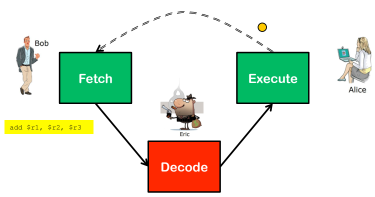
- Taxonomy
  - Ticking Timebombs, Cheat Codes, Emitters, Corrupters
- Solutions
  - TrustNet and DataWatch
- Results
  - Correctness, Coverage and Costs
- Future Work

Sample Emitter Backdoor

- Consider a malicious instruction decoder
  - Decoder emits instructions not in the original program
  - Execution unit faithfully executes them



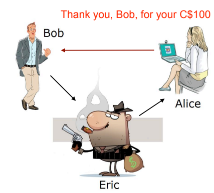
TrustNet



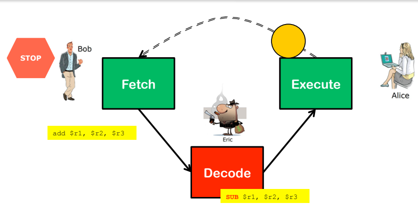
- Predictor and Reactor monitor the Target
  - Division of work prevents *one* bad guy from breaking two units
  - Scaling to larger number increases design complexity

Corrupter Backdoors

- Bob
  - Still nice
  - Donates $100
- Eric
  - Evil (and smarter)
  - Converts to Canadian $
- Alice
  - Still president
  - Fooled by Eric's C$100



DataWatch



- Scaled up version of TrustNet
  - Multiple bit messages
  - Confirms types of messages (instead of just yes/no)

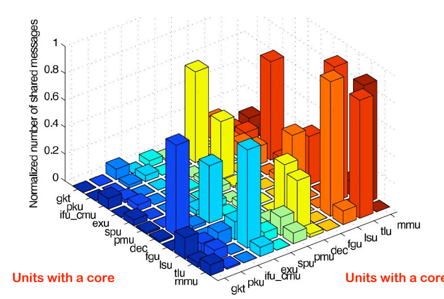
Outline

- Taxonomy
  - Ticking Timebombs, Cheat Codes, Emitters, Corrupters
- Solutions
  - TrustNet and DataWatch
- Results
  - Correctness, Coverage and Costs
- Future Work

Experimental Context, Correctness, Costs

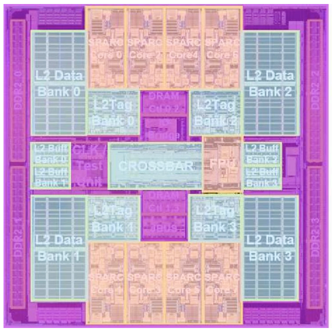
- Context
  - Simplified OpenSPARC T2
- Correctness
  - Designed attacks
  - No false positives or negatives
- Costs
  - Low area overhead (2 KB per core)
  - No performance impact
- How to measure coverage?

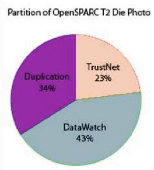
Coverage: Vulnerability Space



Paper has plots for other units at a chip level

Coverage Visualization

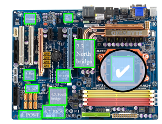




WARNING:
This is an approximate vizualization

Summary and Future Work

- **Strengthen root of trust: microprocessors**
  - Hardware-only solution. No perf impact, low area overhead
  - Security add-on highly resilient to corruption
  - Provided attack taxonomy, method to characterize attack space
- **Applicability of TrustNet & DataWatch**
  - Covered: pipelines, caches and content associative memory
  - Not covered: ALU, microcode, power mgmt., side-channels
- **Moving Forward**
  - Expand coverage
    - Out-of-order processors
    - Motherboard components
  - Design automation tools
  - Reaction to errors
  - Applying techniques for reliable execution



- First steps toward a secure trusted hardware w/ untrusted units

Thank You! and Questions?