Silencing Hardware Backdoors

Adam Waksman and Simha Sethumadhavan

IEEE Symposium on Security and Privacy 2011

Presented by Joshua Schiffman

Systems and Internet Infrastructure Security (SIIS) Lab
Computer Science and Engineering Department
Pennsylvania State University

Turtles All The Way Down

PENNSTATE 1855

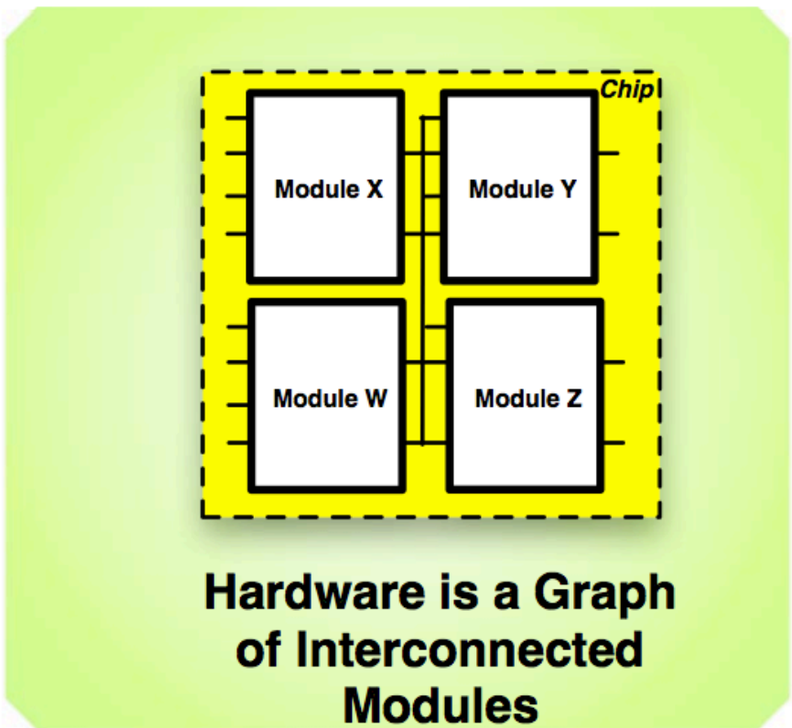
- Integrity of components is based on its TCB
  - Programs depend on other userspace code and the kernel
  - OS Kernel depends on the underlying hardware and its firmware
- Exploits at a lower layer can subvert the higher layers
  - Kernel root kits can defeat virus scanners by lying to them
- What is the lowest layer we can check?
- Reflections on Trusting Trust – Ken Thompson

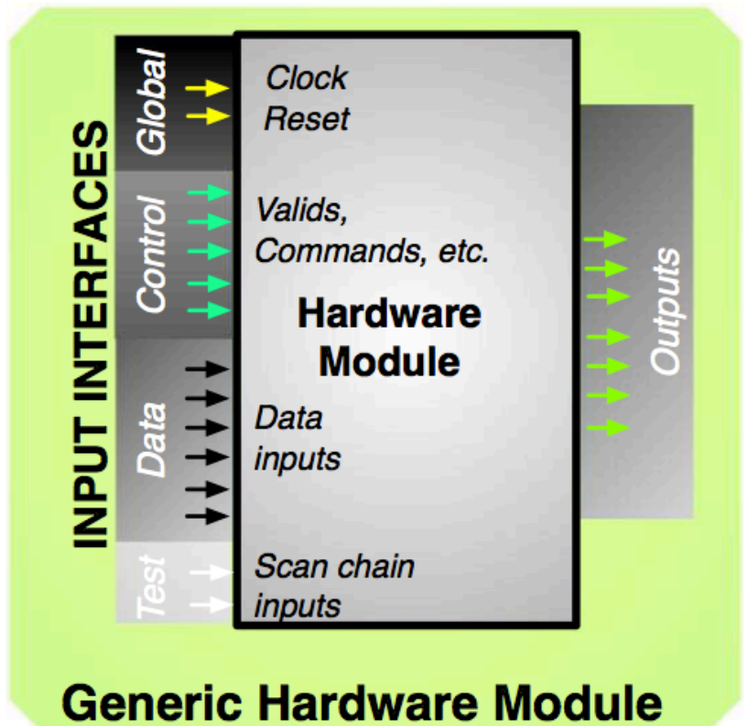
Beyond The Software Boundary

PENNSTATE 1855

- Code can be scanned for malicious logic
- Evil hardware is much harder to find (and stop)
  - Impossible for consumers to inspect
  - Triggers complex enough to bypass hw verification
  - Modern hw uses modules from multiple sources
- Illinois Malicious Processor (King et. al.)
  - Modified a programmable CPU to load backdoor code
  - Triggered by CPU inspecting a network packet
  - Operated in *Shadow Mode* to avoid detection

Model of Hardware





Types of Attacks

- Timebombs: activate after so many cycles or seconds
  - Triggered by global clock input (possibly accumulators)
- Cheat codes: specific sequences trigger the backdoor
  - **Single-shot:** inputs in a single cycle contain the code

    0xDECAFBAD
  - **Sequence:** Code is spread across cycles

    0xD 0xE 0xC 0xA 0xF 0xB 0xA 0xD
- Based on the author's earlier study

Assumptions

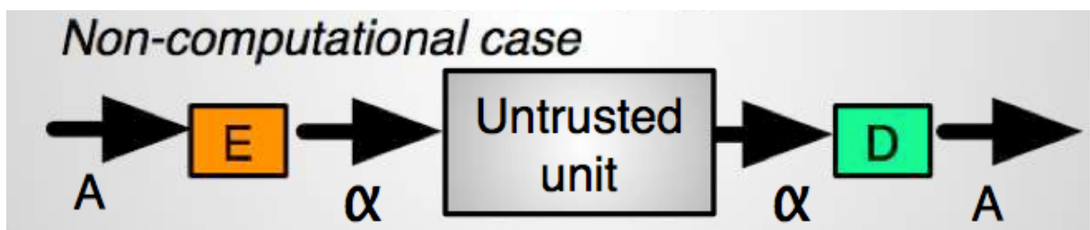
- Triggers are out of scope of validation
  - Validation last millions of cycles
  - Validation input 2^T inputs (T~= 20)
- Validation catches all triggers that are subset of tests
  - Correctly detects erroneous output and superfluous input
- Solution uses small manually verified modules
  - Unprotected, but assumed correct
- No on-chip memory (burn it out)

Solution - Timebombs

- Timebombs triggered by some amount of time
  - Remove guarantee of time
- Random power resets
  - HW circuit resets the module randomly > 2^T cycles
  - Do not touch RAM or large memory structures
  - Use special hw to record performance registers
- What about non-clock-based timing?

Solution - Cheat Codes

- Cheat codes can come on data / control channels
  - Assume control channel is small enough to exhaust
  - Data is either used or bused
- Non-computation modules
  - Encrypt data since module doesn't depend on it
  - Use simple XOR encryption with randomness like PUFs



Solution - Cheat Codes Cont.

- Computational Modules process data
  - Must be able to recover correct results
  - Use homomorphic obfuscation that is easy to reverse
- Homomorphic operations
  - Easy to implement if reversing is simple
  - $x^2y^2 = (xy)^2$
  - RSA(xy) = RSA(X) • RSA(Y)
  - Modules aware of this could modify detect original outputs
  - N-variant duplication as a fallback

Solution - Cheat Codes Seq.

- Sequences: trigger must never reach module
- Solution: Reorder inputs to modules
  - If validation checks all 2^T input sequences, then we must reorder all sequences larger than 2^T
  - Adding physical randomness lowers trigger probability
- Some modules cannot be reordered
  - Use dummy inputs to breakup sequence
  - Can the attacker ignore these?

Evaluation

- Tested their model on a Zesto x86 simulator
  - Added stalls for modules that encrypt / decrypt
  - Modified an SDRAM design to support reorder / insert
- Average overhead with everything was 0.9%
  - Claim concurrent mechanisms overlap
- Highest overhead was 3.4%

Limitations

- Analog circuits: hard to test
- Side-channel induced backdoors
  - What kind of channels would these be?
- How can we design trusted components?

Conclusion

- Hardware backdoors are an insidious threat that is difficult to detect at the consumer and even the design level
- By blocking triggers from reaching the backdoors, we can prevent a large (all?) backdoors from being triggered
- Simulations suggest the impact is minimal, but certain modules and workloads may be adversely affected.

Discussion