Security-Aware Processor Architecture Design
CS 6501 – Fall 2018
Ashish Venkat
Logistics

• Access to Linux Machines
• Gem5 installation
• Benchmarks and Licenses
• Schedule
• Questions on Assignment 1?
• Questions on Assignment 2?
Agenda

• Threat Modeling
• Attack Detection Metrics
• Vulnerability Metrics
• Defense Evaluation
• Open-Source Evaluation Tools
Threat Model

• Know your attacker
  • Identify potential vulnerabilities
  • Define attacker capabilities

• Root of Trust
  • What components of the system can be trusted?
  • Do we have mechanisms in place to verify that they are trusted?

• Environmental Assumptions
  • Software – OS, libraries, etc.
  • Hardware – processor, caches, etc.
  • Interplay with other defenses if assumed.
Attack Detection

• Identify malicious execution patterns that potentially represent attacks
• Trap execution to software or trigger hardware defense
• Mitigates the performance overhead of defenses (Always ON → ON under threat)
• Conventional attack detection strategies:
  • Dynamic Information Flow Tracking (Taint Analysis)
  • Tag-based Authentication of Capabilities (Read CHERI papers)
  • ML-based Malware Signature Detection (Read “On the Feasibility of Online Malware Detection with Performance Counters”)
Dynamic Information Flow Tracking

• General Principle
  • Identify untrusted channels
  • Tag input data from untrusted channels as spurious/tainted
  • Tag the result of a computation as tainted if operands were tainted
  • Restrict information flow of tainted values via a predefined security policy (e.g., restrict jumps to addresses stored in tainted registers)

• Implementation
  • Hardware – Microarchitecture-level all the way down to gate-level
  • Software – Source code, IR, binary code
Attack Detection Metrics

- Detection techniques may result in false positives and true negatives
  - Which is more dangerous?
  - How can you model the accuracy of detection?

Confusion Matrices

Heatmaps
Performance Effects of Attack Detection

• Is attack detection on the critical path?
• Performance penalty due to overtainting
  • Security defense is turned on more frequently
• Performance penalty due to undertainting
  • Incurred if mistaint detection and recovery is built into the system
Identifying and Quantifying Vulnerabilities

• Static Analysis Tools
  • Mine applications for presence of vulnerabilities/exposures that can be exploited (e.g., ROP gadgets, Spectre gadgets, key-dependent memory accesses, etc.)
  • Most online implementations rely on PIN/LLVM-based analysis
  • Many others mine binaries using objdump/IDA-based disassemblers

• Metric: Coverage
  • How many vulnerabilities does our detection/defense mechanism address?
  • How many of these actually exist?
Side Channel Vulnerability Factor

• General Principle
  • Analogous to phase detection in performance evaluation
  • Quantifies the characteristics of the signal decoded by the attacker
  • Compare deltas in victim’s execution patterns (oracle) to those inferred by the attacker (side-channel)
  • High correlation implies greater vulnerability
Side Channel Vulnerability Factor

• Benefits
  • Uses classical statistical methods that are easily adaptable
  • Its differential nature allows for the comparison of different types of signals (e.g., time vs power)
  • Allows for comparison of different defenses and the degree to which they mitigate a broader class of attacks
  • Available at: https://github.com/castl/libsvf
Side Channel Vulnerability Factor

• Pitfalls
  • Requires significant pre-processing and smoothening of signals (see CSV)
  • May not work well when the number of phase changes is small
  • Overkill for targeted defenses that completely eliminate a particular side channel – sometimes it is just easier to show the actual signal rather than computing the correlation.
Defense Evaluation

• Theoretical Methods
  • Formal Verification – Leverage SMT (satisfiability modulo theories) solvers to validate claims under different constraints.
  • Supporting Arguments in Plain English – cover all scenarios (attack/legitimate execution) under different execution constraints.
  • Useful when the outcomes are binary – success/failure.
Defense Evaluation

• Empirical Methods
  
  • Attack Surface Reduction
    
    • Attack surface is loosely defined as the sum of all attack vectors that could be leveraged to mount successful exploits.
    
    • Example: Number of all ROP gadgets relocated by a randomization defense, number of BTB poisoning patterns detected and thwarted.
    
    • Typically represented using stacked bars.
Defense Evaluation

• Empirical Methods
  • Entropy
    • Amount of randomness in the system.
    • Example: Amount of noise introduced into the signal decoded from a side channel, number of manifestations of a ROP gadget, etc.
    • Typically measured as bits.
  • Time to Attack
    • Varies with the amount of entropy
    • Typically measured as number of attempts
Some Open Source Evaluation Tools

• Cache Side Channel Attacks:
  • [https://github.com/IAIK/flush_flush](https://github.com/IAIK/flush_flush)
  • [https://cs.adelaide.edu.au/~yval/Mastik/](https://cs.adelaide.edu.au/~yval/Mastik/)

• Meltdown/Spectre Attacks:
  • [https://github.com/IAIK/meltdown](https://github.com/IAIK/meltdown)
  • [https://github.com/felixwilhelmlm/mario_baslr](https://github.com/felixwilhelmlm/mario_baslr)

• Taint Tracking

• Formal Verification for Hardware Security
  • [https://github.com/ChrisXu1006/SecVerilog](https://github.com/ChrisXu1006/SecVerilog)
  • [https://github.com/cliffordwolf/riscv-formal](https://github.com/cliffordwolf/riscv-formal)
Security-Aware Processor Architecture Design
CS 6501 – Fall 2018
Ashish Venkat