EE 156: Advanced Topics in Computer Architecture

Spring 2023
Tufts University

Instructor: Prof. Mark Hempstead
mark@ece.tufts.edu

Lecture 9: Security Overview and Meltdown/Spectre

Outline

Unit 3: Security from the Hardware/Systems Perspective (2 weeks)

- **Security Principles** [SLCA: R. Lee]
- Secure Architectures and Secure Memory
- Side-Channels and Examples
  - Rowhammer
  - EM (Eddie)
- Hardware Security and Side-Channel Attacks
  - **Spectre and Meltdown**
  - Mitigations

Other Units Planned

Unit 4: Multicore and Heterogeneous Systems (2 weeks)

- Impact of Technology Scaling on Design [Chapter 1]
- Dark Silicon and the End of Technology Scaling
- Data-level Parallelism, SIMD and Vector Architectures [Chapter 4 and G]
- GPU Architectures [Chapter 4]
- Heterogeneous Systems and Many Accelerator Architectures [New H&P Chapter]

Unit 5: Power and Energy (1 week)

- Power Modeling [SLCA: Kaxiras, Martonosi: 2010-Chapters1/2, 2014-Chapter 1]
- Introduction to DVFS [SLCA: Kaxiras, Martonosi: 2010-Chapters 2/3, 2014-Chapter 2]

Security References

- At least in Computer Architecture the field is relatively new and moving quickly. It's not really covered in your textbook
  - Spectre and Meltdown were published Jan 2018
  - Side channel attacks have gained prevalence over the last few years: Timing, Rowhammer, RF/EM, Power and Thermal
- Useful References:
  - Ruby B. Lee Security Basics for Computer Architects Synthesis Lectures on Computer Architecture September 2013 (https://doi.org/10.2200/S00512ED1V01Y201305CAC025)
  - Prof. Simha Sethumadhavan: http://www.cs.columbia.edu/~simha/
  - Prof. Srini Devadas: https://people.csail.mit.edu/devadas/

Caveats

- This unit will not cover all aspects of security
  - Network Security and Intrusion Detection
  - Cryptography
  - Software exploits (buffer overflow etc..)
- We will cover basic principles found in other aspects of security but take a hardware view
  - Hardware can help security by providing trust and hardware support for security (encryption, authentication and security levels)
  - Hardware can be exploited through physical access and physical side channels

General Security Principles

- Three such properties have been called cornerstone security properties (CIA)
  - Confidentiality
  - Integrity
  - Availability

CIA

- **Confidentiality** is the prevention of the disclosure of secret or sensitive information to unauthorized users or entities.
- **Integrity** is the prevention of unauthorized modification of protected information without detection.
- **Availability** is the provision of services and systems to legitimate users when requested or needed. Availability is also a goal in providing reliable, dependable or fault-tolerant systems except that availability, in the security sense, has to also consider intelligent attackers with malevolent intent, rather than just device failures and faults.

Other Important Security Properties and Terminology

Access control	Restrict access to only those allowed to access the information; comprises Authentication and Authorization
Authentication	Determine who a user or machine is
Authorization	Determine what a given subject is allowed to do to a given object
Attribution	The ability to find the real attackers when a security breach has occurred
Accountability	Holding parties (e.g., vendors, operators, owners, users and systems) responsible for (software, hardware, protocol, network or policy) vulnerabilities that enable successful attacks
Audit	Keeping logs to enable re-tracing events and accesses to protected data or services
Attestation	The ability of a system to provide some non-forgeable evidence to a remote user (e.g., of the software stack it is currently running)
Non-repudiation	The ability to ensure that a user cannot deny that he has made a certain request or performed a certain action
Anonymity	The ability to perform certain actions without being identified, tracked or authenticated
Privacy	The right to determine how one's personal information is to be distributed

- General security properties beyond CIA.
- Not all systems have these properties

Threats and Attacks

A **security breach** is an event that violates a security property. For example, it could be a breach of confidentiality, of integrity or of availability, etc. A security breach can be intentionally caused or accidental.

A **threat** is any action that can damage an asset, or cause a security breach. For example, there are disclosure threats, modification threats and denial of service threats, which threaten the mainsecurity goals of protecting confidentiality, integrity and availability, respectively.

An **attack** is a specific instance of a threat. It involves detailed descriptions of the system, the vulnerabilities exploited, the attack path and the assets attacked.

A **vulnerability** is a weakness in a system that can be exploited to cause damage to an asset. One or more vulnerabilities are exploited in an attack.

Example Threats and Attacks

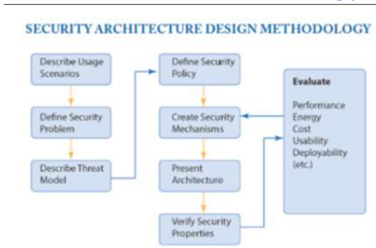
Threats	Examples of Attacks
Confidentiality breaches	Eavesdropping; key-logging; password cracking; secondary dissemination by authorized recipients; leaking information through covert channels or side channels
Integrity breaches	Undetected modifications of data or programs; spoofing attacks; splicing attacks; replay attacks; corrupting audit logs, or other mechanisms for attribution, accountability and security
Availability breaches	Packet dropping; Denial of Service (DoS) attack; distributed Denial of Service attack (DDoS); network flooding; resource depletion
Authentication threats	Masquerading, impersonation; identity theft
Privacy breaches	Leaking sensitive personal information, e.g., medical record, salary, browsing habits, etc.
Anonymity breaches	Tracing tools that can accurately identify an individual, entity or machine
Insider threats	Attacker is an authorized person who knows the system well, e.g., a software or hardware designer, system administrator; covert channel attacks
Hardware attacks	Physical attacks; attacks on hardware; side-channel attacks; hardware Trojans
Software attacks	Attacks on software; operating system attacks; hypervisor attacks; API attacks; web attacks, e.g., cross-site scripting attacks; malware, e.g., viruses, worms and Trojans
Protocol attacks	Man-in-the-middle attacks; reflection attacks; replay attacks; IP spoofing attacks
Network-enabled threats	Virus attacks; worm attacks; DDoS attacks; firewall attacks; intrusions; protocol attacks
Other types of attacks	Byzantine attacks; script kiddie attacks

Threat Based Design

- A **threat model** specifies the types of threats that the system defends against, and which threats are not considered.

  For example, in certain usage scenarios, preventing confidentiality breaches is essential while availability is not specifically considered. This means that even if a Denial of Service attack occurs, the sensitive information must not be disclosed to an attacker. For example, the confidential information may be encrypted with the encryption key unavailable to the attacker, or the information may be automatically erased upon an attack.

R. Lee's Secure Design Methodology



Other Topics in R. Lee's Text

- Security Policy Models and Multi-level security (defining different layers of security for different information or users)
- Access Control and Authentication
  - Passwords, Biometrics, Private Keys
- Cryptography for Confidentiality and Integrity
  - Common ciphers
  - Common Hash functions
- Public-key Cryptography (RSA and PKI)
- Security Protocols

Spectre and Meltdown: Recent Security Flaws

Slides from Prof. Srini Devadas, MIT
Given at BARC
January 26th, 2018

ALSO Includes slides from his HPCA 2019 Keynote

Architectural Isolation of Processes

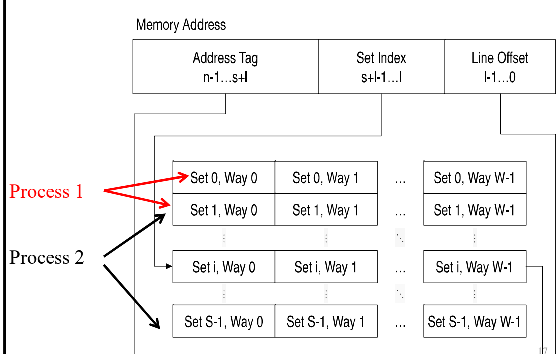
Fundamental to maintaining correctness and privacy!

Performance Dictates Microarchitectural Optimization

Isolation Breaks Because of Shared Microarchitectural State!

16

Shared Last Level Cache



17

Control Flow Speculation is insecure

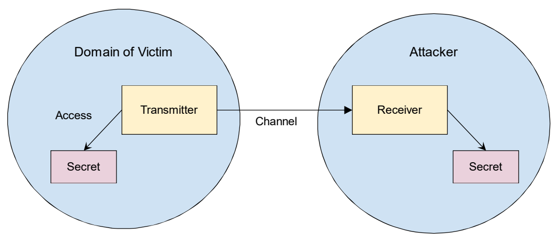
Speculative execution does not affect architectural state → "correct"
… but can be observed via some "side channels" (primarily cache tag state)

… and attacker can influence (mis)speculation (branch predictor inputs not authenticated)

A huge, complex attack surface!

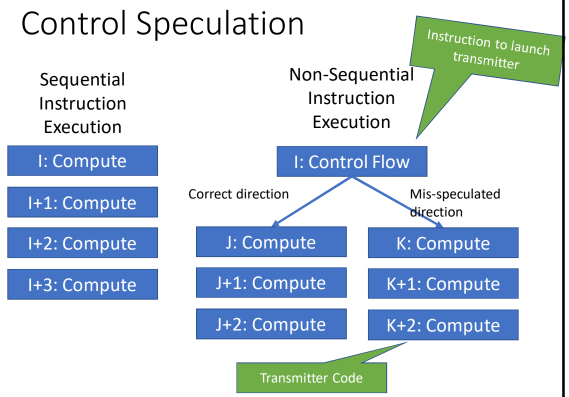
18

Attack Schema

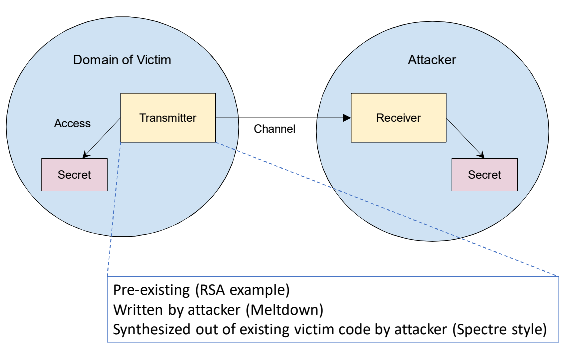


1. Create a channel
2. Create the transmitter
3. Launch the transmitter
4. Access the secret

Control Speculation

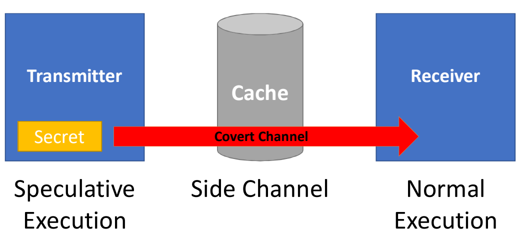


Building a Transmitter



Meltdown and Spectre Attack Examples

Attack: Mis-speculation exfiltrates secrets through cache



Meltdown

Problem: Attacker can influence speculative control flow

Bug: Speculative execution not subject to page permission checks

Attack: User code can read kernel data (secret)

Three steps:

1. Setup: flush the cache
2. Transmit: force speculation that depends on secret
3. Receive: measure cache timings

Meltdown example

Setup:

```
clflush(timing_ptr[guess]);
```

Transmit:

```
timing_ptr[*kernel_addr];
```

Page Fault

May still read *kernel_addr (speculatively)

Receive:

```
mfence();
s = rdtsc(); *timing_ptr[guess];
e = rdtscp();
if (e - s < CACHE_MISS_THRESHOLD)
  printf("guess was right!\n");
```

Spectre

- Problem: Attacker can influence speculative control flow (same as before)
- Attack: Exfiltrate secrets within a process address space (e.g. a web browser). Can also be used to attack the kernel.
- Could use attacker provided code (JIT) or could co-opt existing program code
- Same three steps! Different setup and transmitters.

Spectre examples

Transmit - Branch Target Injector:

```
fnptr_t foo = choose_function();
foo(bar);
```

Transmit - Bounds Check Bypass:

```
if (x < array1_size)
  array2[array1[x] * 256];
```