Cache Side Channels:

State of the Art and Research Opportunities

Yinqian Zhang, Ph.D.
Assistant Professor
The Ohio State University

Self Introduction

- Research interests
  - Computer system security, (micro-architectural) side-channel attacks and defenses
- Recent publications on side channels
  - Cloud computing (S&P'11, CCS'12, CCS'13, CCS'14, Security'15, Security'16, RAID'16, CCS'16a, AsiaCCS'17b)
  - Smartphones (CCS'15, CCS'16b, NDSS'18a)
  - Intel SGX (AsiaCCS'17a, CCS'17a, CCS'17b)
- Fortunate to served on the following conference PCs in the past 3 years
  - IEEE S&P: 2016, 2017, 2018
  - ACM CCS: 2015, 2016, 2017
  - USENIX Security: 2017
  - NDSS: 2017, 2018

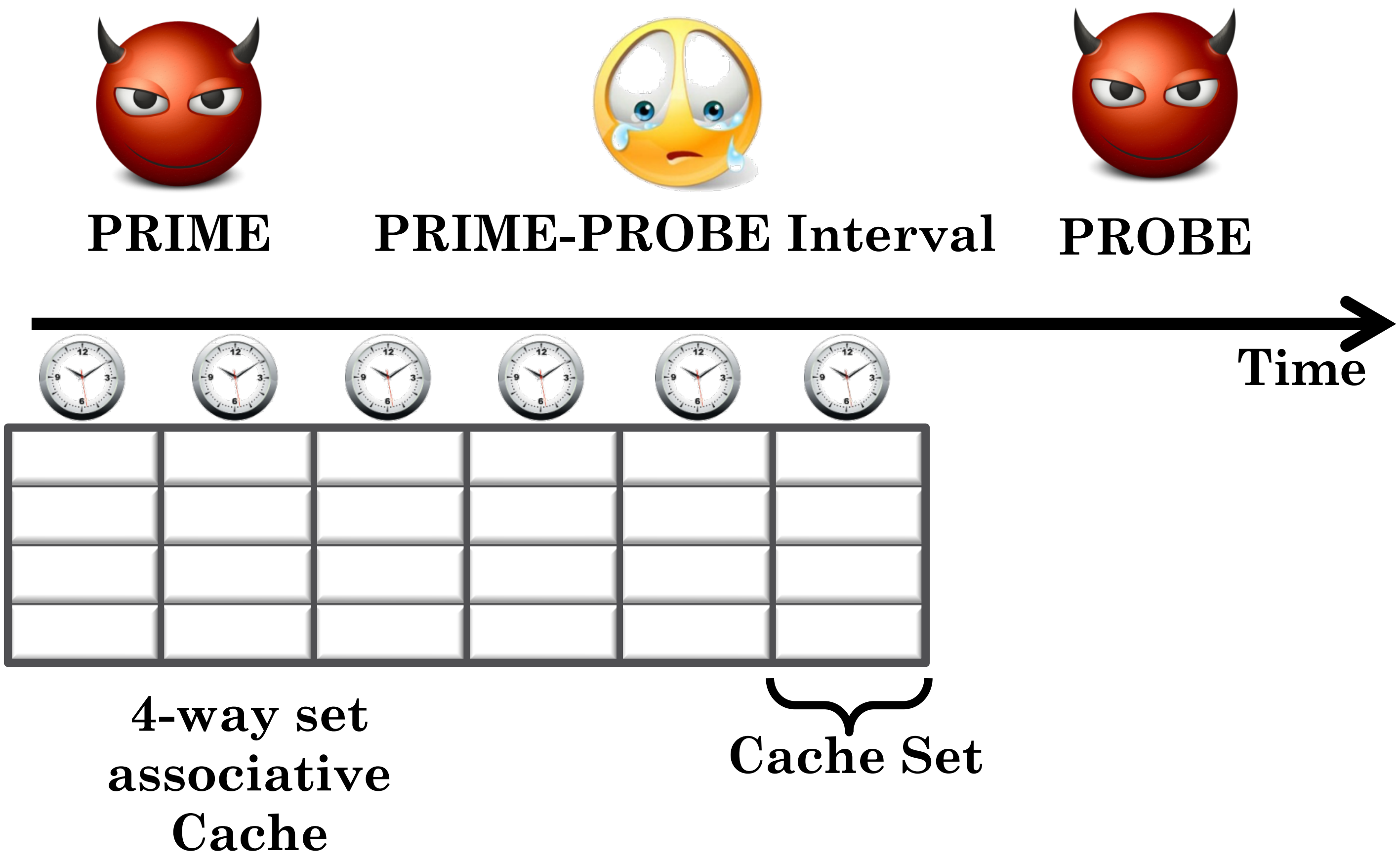
Cache Side-Channel Attacks

The Basics

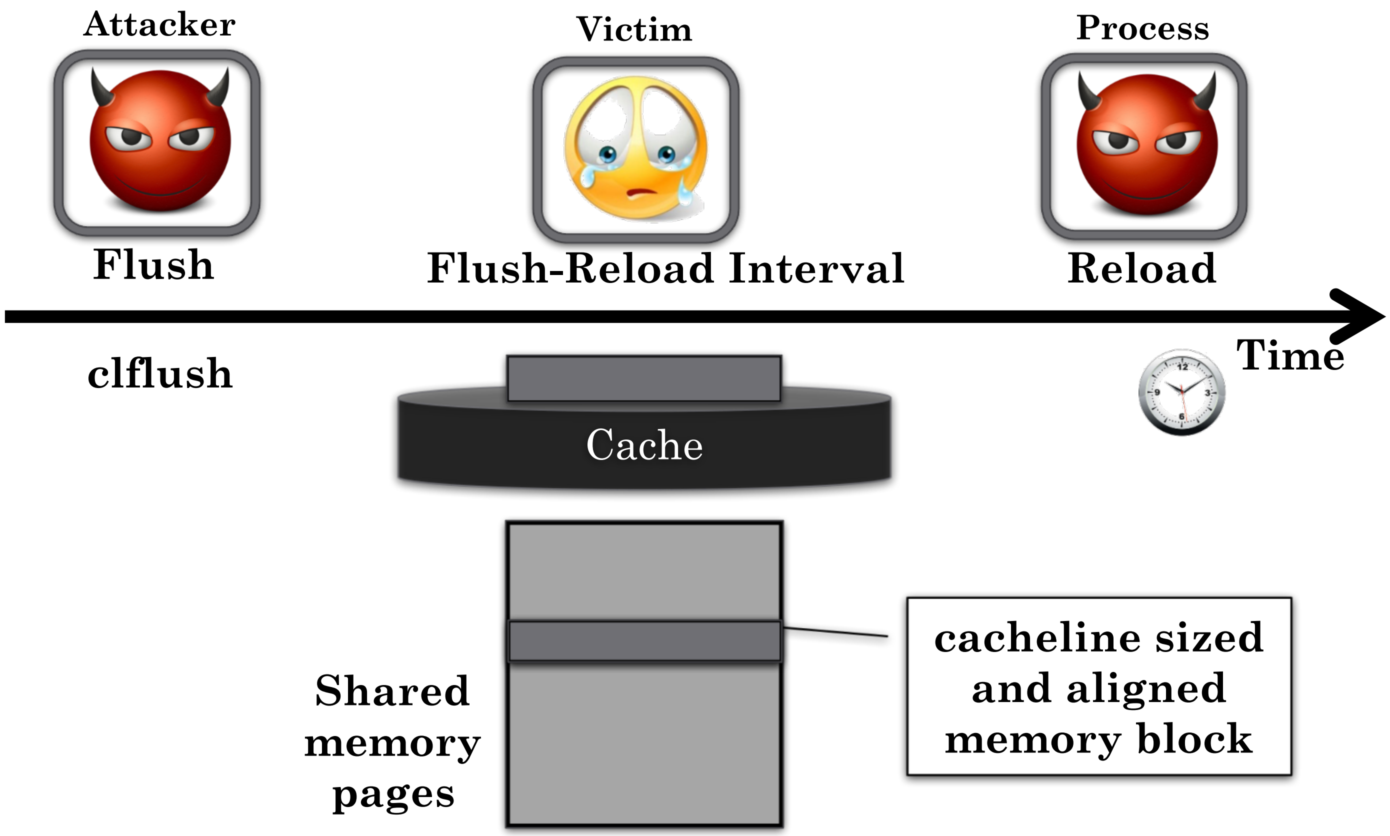
Threat Models

- Cache side-channel attacks
  - Time driven
  - Trace driven
  - Access driven
- Access-driven cache attacks
  - Logical accesses to the target computer system
  - Share cache(s) with the victim program
  - Attacker accesses its own memory region (and time the accesses) to infer victim's use of the shared cache
    - Evict+Time
    - Prime+Probe
    - Flush+Reload
    - ……

Prime+Probe Attacks



Flush+Reload Attacks



Other Attack Techniques

- Evict+Time Attacks
  - Attacker evicts one or more cache sets
  - Attacker measures the total execution time of a cryptographic operation
- Flush+Flush Attacks
  - Similar to Flush+Reload attacks
  - The second Flush to replace Reload in the Flush+Reload attacks
- Prime+Abort Attacks
  - Leverage hardware transaction memory
  - Use transaction aborts to replace timing

Taxonomy of Cache Side-Channel Attacks

- Shared cache sets
  - Attacker and victim share the same cache set(s)
  - In physically-indexed cache (e.g., last-level cache) attacks, attacker needs to know virtual-to-physical mapping of the victim
  - Example: Prime+Probe, Prime+Abort
- Shared cache lines
  - Attacker and victim share the same cache lines
  - Attacker needs to share some physical memory pages with the victim
  - Example: Flush+Reload, Flush+Flush

Agenda

- Research directions in cache side-channel attacks
  - From same core to cross core
  - From x86 to ARM
  - New attack techniques
  - Beyond cryptographic attacks
  - Non-native code attacks
  - Attacks against strong isolation

- Research directions in cache side-channel defenses
  - Cache partition
  - Access randomization
  - Removing high-resolution timers
  - Runtime attack detection
  - Patching vulnerable programs

Research Direction 1

From Same Core to Cross Core

From Same-core Attacks to Cross-core Attacks

- Single-core processors
  - Simultaneous multi-threading (SMT)
    - Intel Pentium 4 (Hyper-Threading): 2002
- Multi-core processors
  - Intel Pentium D: 2005
  - AMD Athlon 64 X2: 2005
- Inclusive last-level caches
  - Intel Nehalem: 2008
- Non-inclusive last-level caches
  - Skylake-SP processors 2017 (Core i9)

2005: SMT-based L1 cache attacks

2014: cross-core Flush+Reload attacks

2015: cross-core Prime+Probe attacks

Existing Studies (1)

- Yarom and Falkner, *FLUSH+RELOAD: a High Resolution, Low Noise, L3 Cache Side-Channel Attack,* USENIX Security 2014.

- Flush-Reload Attacks on last-level caches using the *clflush* instruction.
  - First invention of the name Flush+Reload Attacks
  - Same-core Flush+Reload attacks was invented before:
    - D. Gullasch and E. Bangerter and S. Krenn, Cache games -- Bringing access-based cache attacks on AES to practice, IEEE S&P 2011.
- A fine-grained channel that requires memory sharing between the two parties
  - Finer-grained than Prime+Probe attacks
  - Requires inclusive cache to propagate cache invalidation (clflush) to other cores
  - A number of follow-up works

Existing Studies (2)

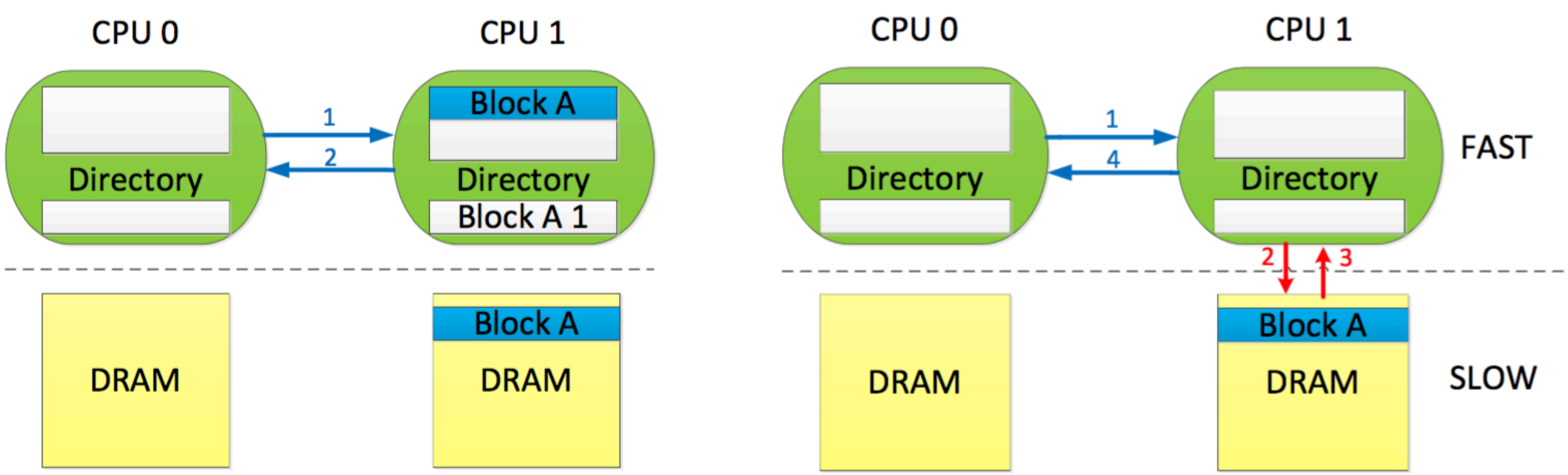
- Liu, Yarom, et al., *Last-Level Cache Side-Channel Attacks are Practical*, IEEE S&P, 2015.
- Irazoqui et al., *S$A: A Shared Cache Attack That Works across Cores and Defies VM Sandboxing -- and Its Application to AES*, IEEE S&P, 2015.

- Prime+Probe attacks on last-level caches by taking advantage of cache inclusiveness
  - Prime: Cache line eviction in the LLC also invalidates other per-core caches
  - Probe: Memory accesses from other cores will miss in their private caches, thus also affects the shared LLC

Existing Studies (3)

- Irazoqui et al., *Cross Processor Cache Attacks*, ASIACCS 2016.

- Cross-CPU Flush+Reload attacks by leveraging cache coherence protocols



*Figures copied from the original paper.

Open Research Questions

- New micro-architecture design features require new side-channel attack designs
  - Cache line replacement policy (LRU, random, adaptive policies)
  - LLC: inclusive, non-inclusive, exclusive
  - Cache internal structure: L1 cache banks, LLC slices
  - Implementation of cache line invalidation instructions, e.g., clflush
  - Cache coherence control.

Research Direction 2

From x86 to ARM

Cache Side-Channel Attacks on ARM

- Targets of ARM cache attacks:
  - Mobile devices (e.g., Android, iOS)
  - ARM-powered data centers
- Challenges:
  - Unclear ARM specifications (and whether they are strictly followed on a specific chip)
  - Unclear processor implementation details
    - Cache line replacement policy
    - Cache inclusiveness
    - Implementation of cache line invalidation instructions
    - Cache coherence control.
  - Difference in the instruction set architecture (compared to x86)

Existing Studies

- Lipp et al., *ARMageddon: Cache Attacks on Mobile Devices*, USENIX Security 2016.
  - Prime+Probe, Flush+Reload, Evict+Reload attacks
- Zhang et al., *Return-Oriented Flush-Reload Side Channels on ARM and Their Implications for Android Devices,* ACM CCS 2016.
  - Flush+Reload attacks
- Green et al., *AutoLock: Why Cache Attacks on ARM Are Harder Than You Think*, USENIX Security 2017.
  - An undocumented autolock mechanism that affects Prime+Probe attacks

Open Research Questions

- Understanding of the attack vectors
  - Conflicted research results (even on the same types of devices)
  - Lack of ground truth (ARM specification?)

- Demonstration of attacks that matter
  - Need a compelling example

Research Direction 3

New Attack Techniques

New Cache Side-Channel Attack Techniques

- 2005: L1 cache Prime+Probe and Evict+Time attacks using SMT
- 2007: L1 cache Prime+Probe attacks without SMT
- 2010: L1 cache Flush+Reload attacks without SMT
- 2014: Cross-core Flush+Reload attacks
- 2015: Cross-core Prime+Probe attacks
- 2016: Cache storage-channel attacks
- 2016: Cross-core Flush+Flush attacks
- 2017: Cross-core Prime+Abort attacks
- 2017: Side channels leveraging Intel Processor Trace
  - Need to be performed with kernel privileges

Existing Studies (1)

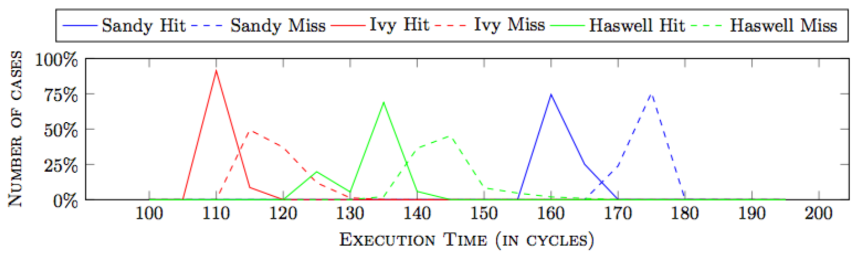
- Guanciale et al., *Cache Storage Channels: Alias-Driven Attacks and Verified Countermeasures*, IEEE S&P 2016.

- Root cause of the side channels
  - Accessing the same physical address through virtual aliases with mismatched cacheability attributes.
  - Executing self-modifying code without flushing the instruction cache
- Enabling Prime+Probe cache attacks without timers
  - Extracting 128-bit key from an AES encryption service running in TrustZone
  - Subverting modular exponentiation in the same platform

Existing Studies (2)

- Gruss et al., *Flush+Flush: A Fast and Stealthy Cache Attack,* DIMVA 2016

- Measure the execution time of the second Flush
  - Key insight: clflush executes faster if cache hit



- Compared to Flush+Reload attacks
  - Lower execution time than Flush
  - Flush+Flush attacks are not detectable by hardware performance counters
    - Reload typically induce a large number of cache misses

Existing Studies (3)

- Disselkoen et al., *Prime+Abort: A Timer-Free High-Precision L3 Cache Attack using Intel TSX*, USENIX Security 2017.

- Use Intel Transactional Synchronization Extensions (TSX) to monitor cache line eviction
  - Transaction aborts if cache lines in write-set or read-set are evicted
  - L1 Prime+Abort: with SMT
  - LLC Prime+Abort
- Key differences from Prime+Probe
  - Timer-less attacks
  - Less noisy
  - Slightly less information (Prime+Abort is binary)

Existing Studies (4)

- Lee et al., *Inferring Fine-grained Control Flow Inside SGX Enclaves with Branch Shadowing*, USENIX Security 2017.

- Main contribution: demonstration of BTB side channel attacks on SGX
- Use Intel Processor Trace to measure timing between branch instructions
  - Need system privilege – only useful in SGX side-channel attacks
  - Similarly, hardware performance counters have been demonstrated to replace timers
    - But SGX does not allow HPC in enclave mode

Open Research Questions

- Incremental improvements
  - Reduce noise
  - Improve accuracy, robustness
- Significant improvements
  - New techniques for cache side channels
  - Addressing some limitations of previous attacks
  - Challenge existing defenses

Research Direction 4

Beyond Cryptographic Attacks

Targets of Cache Side-Channel Attacks

- Cryptographic attacks
  - Modular exponentiation (RSA): Square-and-multiply
  - Key dependent table accesses (AES): s-box
  - Scalar multiplication (ECDSA) : double-and-add
- User privacy
- Address space layout randomization (ASLR)
  - JavaScript code infer browser user space ASLR
  - Native code infer kernel space ASLR (KASLR)

Existing Studies (1)

- Oren et al., *The Spy in the Sandbox: Practical Cache Attacks in JavaScript and their Implications*, ACM CCS 2015.
  - Tracking user behavior
    - e.g., proximity sensor
- Zhang et al., *Return-Oriented Flush-Reload Side Channels on ARM and Their Implications for Android Devices,* ACM CCS 2016.
  - Detecting hardware events
    - e.g., touchscreen interrupts
  - Tracing software execution path
    - e.g., push notification, display updates

Existing Studies (2)

- Gras et al., *ASLR on the Line: Practical Cache Attacks on the MMU*, NDSS 2017.

- Malicious JavaScript code de-randomizes the layout of the browser's address space, solely by accessing memory
- Key techniques:
  - Prime+Probe and Evict+Time attacks to infer page table accesses after a page walk
  - To address coarse-grained performance.now()
    - Time to tick: performance.now() until tick
    - Shared memory counter: A web worker thread to create a software clock

Open Research Questions

- What other secrets might be vulnerable to cache side channels?
  - Secret-dependent memory accesses
  - Text data itself is usually not a target
- High-impact targets will advance the research field
  - Software/hardware vendors' attention will motivate invention and adoption of defenses

Research Direction 5

From Native Code to JavaScript

JavaScript Cache Side-Channel Attacks

- Unprivileged JavaScript code running in browsers
- Oren et al., *The Spy in the Sandbox: Practical Cache Attacks in JavaScript and their Implications*, ACM CCS 2015.
  - Prime+Probe attacks using JavaScript
    - Constructing cache eviction set (using JavaScript code)
  - Timer: performance.now()
- Gras et al., *ASLR on the Line: Practical Cache Attacks on the MMU*, NDSS 2017.
  - Prime+Probe and Evict+Time attacks to infer page table accesses after a page walk
  - Timer: Timing to tick or shared memory counter in a JavaScript web worker

Open Research Questions

- Attacks from other non-native languages
  - Challenges:
    - Lack of clflush instructions
    - Creating eviction buffers
    - High-resolution timers
  - Example scenarios
    - Java
    - JavaScript in non-browser settings
- Attacks against non-native languages
  - Challenges:
    - Memory management in the runtime is complex
  - Example scenarios
    - Managed cloud applications, PaaS, Microservice, etc.

Research Direction 6

Attacks against Strong Isolation

Attacks against Strong Isolation

- Virtualization and cloud computing
  - Same-core attacks
  - Cross-core attacks
- Trusted Execution Environments
  - SGX side-channel attacks
  - TrustZone side-channel attacks

Cross-VM Side-Channel Attacks

- Prime+Probe side-channel attacks
  - Same-core attacks
    - Zhang et al., *Cross-VM Side Channels and Their Use to Extract Private Keys*, ACM CCS 2012
  - Cross-core attacks
    - Liu et al., *Last-Level Cache Side-Channel Attacks are Practical*, IEEE S&P, 2015.
    - Irazoqui et al., *S$A: A Shared Cache Attack That Works across Cores and Defies VM Sandboxing -- and Its Application to AES*, IEEE S&P, 2015.
    - Inci et al., *Seriously, get off my cloud! Cross-VM RSA Key Recovery in a Public Cloud*, 2015
- Flush+Reload side-channel attacks
  - Requires cross-VM memory deduplication
  - Existing studies
    - Yarom and Falkner, *FLUSH+RELOAD: a High Resolution, Low Noise, L3 Cache Side-Channel Attack,* USENIX Security 2014.
    - Yarom and Benger, *Recovering OpenSSL ECDSA Nonces Using the FLUSH+RELOAD Cache Side-channel Attack,* IACR eprint, 2014
    - Irazoqui et al., *Fine Grain Cross-VM Attacks on Xen and Vmware*, BDCLOUD, 2014

SGX Side-Channel Attacks

- L1 cache Prime+Probe side-channel attacks with SMT
  - Brasser et al., *Software Grand Exposure: SGX Cache Attacks Are Practical*, USENIX Workshop on Offensive Technologies (WOOT), 2017
- LLC Prime+Probe side-channel attacks
  - Schwarz et al., *Malware Guard Extension: Using SGX to Conceal Cache Attacks*, Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA), 2017
- L1 cache Prime+Probe side-channel attacks with interrupts
  - Hähnel et al., *High-Resolution Side Channels for Untrusted Operating Systems*, USENIX ATC, 2017

TrustZone Side-Channel Attacks

- Zhang et al., *TruSpy: Cache Side-Channel Information Leakage from the Secure World on ARM Devices,* https://eprint.iacr.org/2016/980.pdf
  - Cache Prime+Probe attacks against TrustZone secure world
    - Attackers may be a kernel module in the normal world or an Android app
    - A single core CortexA-8 processor on a Freescale i.MX53 development board

- Guanciale et al., *Cache Storage Channels: Alias-Driven Attacks and Verified Countermeasures*, IEEE S&P 2016.
  - Prime+Probe attacks without timers
    - Accessing the same physical address through virtual aliases with mismatched cacheability attributes.
    - Executing self-modifying code without flushing the instruction cache

Open Research Questions

- Cache side-channel attacks to break stronger security isolation has motivated this research field in the past few years.
- Cloud computing
  - Attacks demonstrated in public clouds already
  - Need stronger evidence to demonstrate the practicality of the attacks
- Trusted Execution Environment
  - Cache attacks against SGX is well studied; targeting known vulnerable software is less interesting
  - A real-world cache side-channel attack against TrustZone is missing

Cache Side-Channel Defenses

Direction 1: Cache Partition

Hardware Solutions

- New hardware designs to partition cache
  - Redesign of CPU caches
  - Simulation for performance evaluation
  - Adoption by CPU vendors is difficult
- Existing Studies
  - Wang and Lee, *New cache designs for thwarting software cache-based side channel attacks*, ISCA 2007
  - Wang and Lee, *A novel cache architecture with enhanced performance and security,* MICRO 2008
  - Domnitser et al., *Non-monopolizable caches: Low-complexity mitigation of cache side channel attacks*. ACM Trans. Archit. Code Optim. 8, 4 (Jan. 2012)
  - Kong et al., *Architecting Against Software Cache-Based Side-Channel Attacks*. IEEE Trans. Comput. 62, 7 (July 2013).

System-level Spatial Partition

- Key ideas
  - Statically or dynamically partition the shared caches by modifying operating systems or hypervisors
- Existing Studies
  - Raj et al., *Resource Management for Isolation Enhanced Cloud Services*. ACM CCSW 2009.
  - Shi et al., *Limiting cache-based side-channel in multi-tenant cloud using dynamic page coloring*. DSN-W 2011.
  - Kim et al., *STEALTHMEM: system-level protection against cache-based side channel attacks in the cloud*. USENIX Security 2012.
  - Zhou et al., *A Software Approach to Defeating Side Channels in Last-Level Caches*. CCS 2016.
  - Liu et al., *CATalyst: Defeating Last-Level Cache Side Channel Attacks in Cloud Computing*. HPCA 2016.

System-level Temporal Partition

- Key ideas
  - Cleanse caches upon context switch
  - Disallow shared use of resources
- Existing Studies
  - Zhang and Reiter, *Düppel: Retrotting Commodity Operating Systems to Mitigate Cache Side Channels in the Cloud*. CCS 2013.
  - Varadarajan et al., *Scheduler-based Defenses against Cross-VM Side-channels*. USENIX Security 2014.
  - Zhou et al., *A Software Approach to Defeating Side Channels in Last-Level Caches*. CCS 2016.

Open Research Questions

- Hardware solutions
  - Better design of cache coherence protocols, last-level cache inclusiveness, and effect of cache invalidation instructions
- System-level solutions
  - Solutions in cloud computing has been broadly studied
    - Need solutions that work well with the cloud business model
  - Scenarios like mobile OS or browsers are less explored
    - cache partition for JavaScript code
    - Android-level cache partition

Cache Side-Channel Defenses

Direction 2: Access Randomization

Hardware Solutions

- New hardware design to introduce randomization in cache uses
  - Randomizing cache line replacement
- Existing studies
  - Wang and Lee. *Covert and Side Channels Due to Processor Architecture*. ACSAC 2006.
  - Wang and Lee, *New cache designs for thwarting software cache-based side channel attacks*, ISCA 2007
  - Wang and Lee, *A novel cache architecture with enhanced performance and security,* MICRO 2008
  - Keramidas et al. *Non Deterministic Caches: A Simple and Effective defense against side channel attacks.* Design Automation for Embedded Systems (2008).
  - Liu and Lee. Random Fill Cache Architecture. MICRO 2014.
  - Liu et al. *GhostRider: A Hardware-Software System for Memory Trace Oblivious Computation,* ASPLOS 2015.

Software Solutions

- Compiler assisted approach to transform applications to randomize its memory access patterns.
- Existing Studies
  - Liu et al. *GhostRider: A Hardware-Software System for Memory Trace Oblivious Computation,* ASPLOS 2015.
  - Crane et al. *Thwarting Cache Side-channel Attacks through Dynamic Software Diversity*. NDSS 2015.
  - Rane et al. *Raccoon: Closing Digital Side-Channels through Obfuscated Execution*. USENIX Security 2015

Open Research Questions

- Leveraging randomness for side-channel protection needs further investigation
  - Randomness may be a target of side channels
  - Entropy-based evaluation?
- More studies are warranted in this direction

Cache Side-Channel Defenses

Direction 3: Removing High-Resolution Timers

Removing High-Resolution Timers

- Hardware solutions
  - Martin et al. *TimeWarp: Rethinking Timekeeping and Performance Monitoring Mechanisms to Mitigate Side-Channel Attacks*. ISCA 2012.
- Hypervisor solutions
  - Aviram et al. *Determinating Timing Channels in Compute Clouds*. CCSW 2010.
  - Vattikonda et al. *Eliminating Fine Grained Timers in Xen*. CCSW 2011
  - Li et al. *StopWatch: A Cloud Architecture for Timing Channel Mitigation*, DSN 2013
- Browser solutions
  - Kohlbrenner and Shacham, *Trusted Browsers for Uncertain Times*, USENIX Security 2016.
  - Cao et al. Deterministic Browser, CCS 2017

Cache Side-Channel Defenses

Direction 4: Runtime Attack Detection

Runtime Attack Detection

- System-assisted side-channel attack detection (for Cloud)
  - Demme et al., *On the Feasibility of Online Malware Detection with Performance Counters*. ISCA 2013.
  - Zhang et al., *CloudRadar: A Real- Time Side-Channel Attack Detection System in Clouds*. RAID 2016.
- Compiler-assisted side-channel attack detection (for SGX)
  - Shih et al., *T-SGX: Eradicating Controlled-Channel Attacks Against Enclave Programs*, NDSS 2017.
  - Chen et al., *Detecting Privileged Side-Channel Attacks in Shielded Execution with DÉJÀ VU*, ASIACCS 2017.
  - Gruss et al., *Strong and Efficient Cache Side-Channel Protection using Hardware Transactional Memory*. USENIX Security 2017.

Open Research Questions

- Reducing performance overhead of runtime detection
  - How to apply detection systems in cloud computing
  - Will cloud providers adopt the technology?
- Attack detection systems in other scenarios
  - Browser? Mobile devices?
- Security policies upon side-channel attack detection
  - What to do after detection?
  - False detection rate?

Cache Side-Channel Defenses

Direction 5: Patching Vulnerable Programs

Existing Software Solutions

- Eliminating side-channel vulnerabilities
  - Molnar et al. *The Program Counter Security Model: Automatic Detection and Removal of Control-Flow Side Channel Attacks*. 2005
  - Coppens et al. *Practical Mitigations for Timing-Based Side-Channel Attacks on Modern x86 Processors*. IEEE S&P 2009.
  - Shinde et al. *Preventing Page Faults from Telling Your Secrets*, ASIACCS 2016.
- Detecting side-channel vulnerabilities
  - Doychev et al., *CacheAudit: A tool for the static analysis of cache side channels*. USENIX Security 2013.
  - Wang et al., CacheD: Identifying Cache-Based Timing Channels in Production Software. USENIX Security 2017.
  - Xiao et al., *Stacco: Differentially Analyzing Side-Channel Traces for Detecting SSL/TLS Vulnerabilities in Secure Enclaves*. ACM CCS 2017.

Open Research Questions

- Neither vulnerability detection nor elimination is completely solved
- New tools are still needed
  - Compiler-assisted solutions (with source code)
  - Binary rewriting (without source code)
- Leveraging program analysis techniques
  - Static analysis: improve accuracy
  - Dynamic analysis: improve coverage

Cache Side Channels: Research Directions

- Research directions in cache side-channel attacks
  - From same core to cross core
  - From x86 to ARM
  - New attack techniques
  - Beyond cryptographic attacks
  - Non-native code attacks
  - Attacks against strong isolation

- Research directions in cache side-channel defenses
  - Cache partition
  - Access randomization
  - Removing high-resolution timers
  - Runtime attack detection
  - Patching vulnerable programs

Thank You!

yinqian@cse.ohio-state.edu