Intro to Microarchitectural Attacks

Thomas Eisenbarth

12.06.2018

Summer School on Real-World Crypto & Privacy

Šibenik, Croatia

UNIVERSITÄT ZU LÜBECK
STIFTUNGSUNIVERSITÄT
SEIT 2015

WPI

Outline

- Timing Attacks
- Cache Attacks
- Cloud Cache Attacks
- Speculative Execution Attacks
- Preventing Microarchitectural Attacks

Timing attack on Password

- Password check done symbol by symbol:

```
def check_pwd(input, pwd):
    for idx in range(len(pwd)):
        if pwd[idx]!=input[idx]:
            return false
    return true
```

- Wrong character results in immediate error message → **Timing dependency**
- **Divide and Conquer approach** allows password recovery in linear time

Timing Attacks

- Password Timing Example:

$$time = f(\color{#1f3864}{input}, \color{red}{secret})$$

- Applied to crypto implementations by Paul Kocher: Diffie-Hellman, RSA, DSS [Koch96]
- Leakage exists, how to exploit it?
  - predict secret dependent timing variations
  - timing differences allow piece-wise key recovery
- **Prevention:** *Write constant-time code*

[Koch96] Paul C. Kocher: Timing attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems - Crypto 96

Microarchitectural Attacks

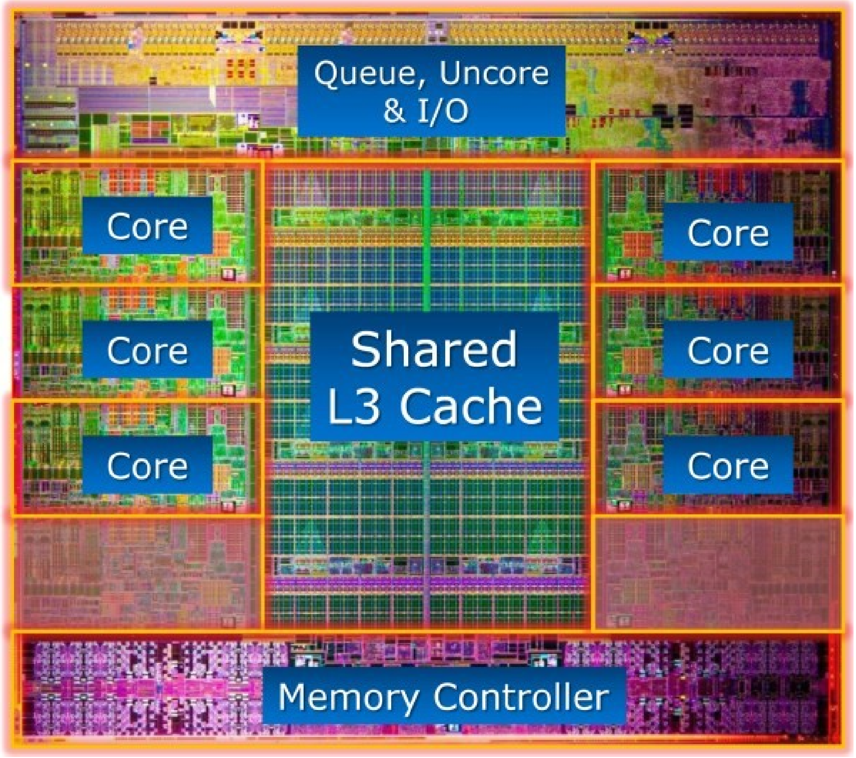
Microarchitectural Attacks

or how to hide secrets in execution time

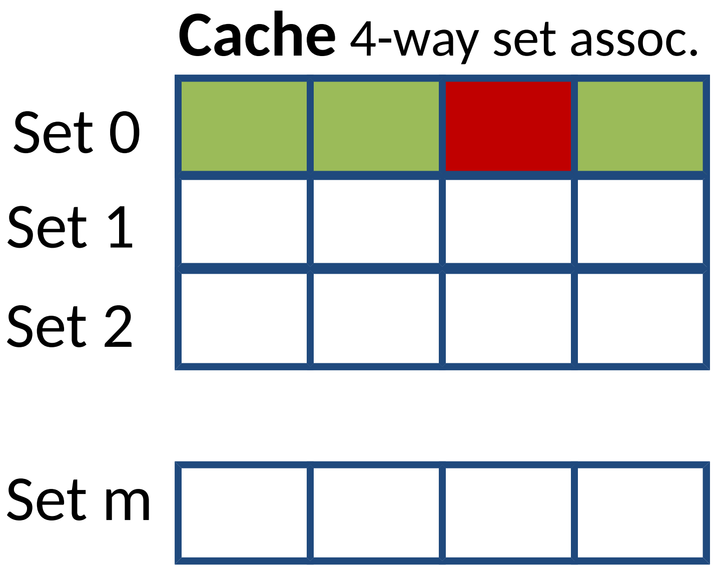
Modern CPUs microarchitecture:

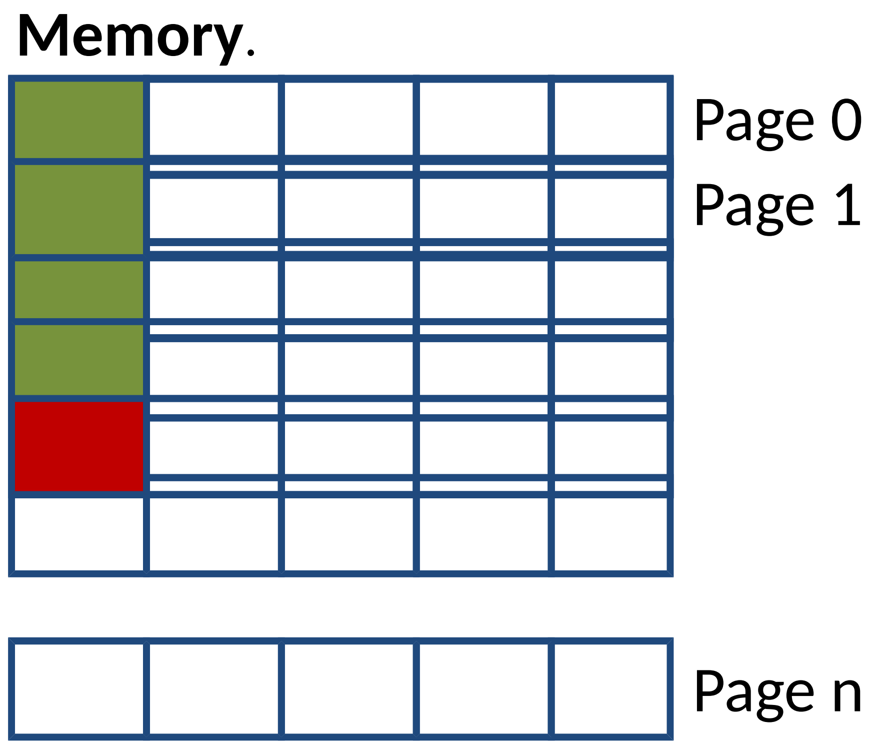
"Make the common case fast"

- Branch Prediction
- Speculative & Out of Order Execution
- Multicore + Multi-processor System & Support
- Several layers of **Caches**



Cache lines and line placement





line size: 64 bytes

Physical Memory Address determines placement in set

Eviction Set: Lines filling one set entirely

Cache Attacks?

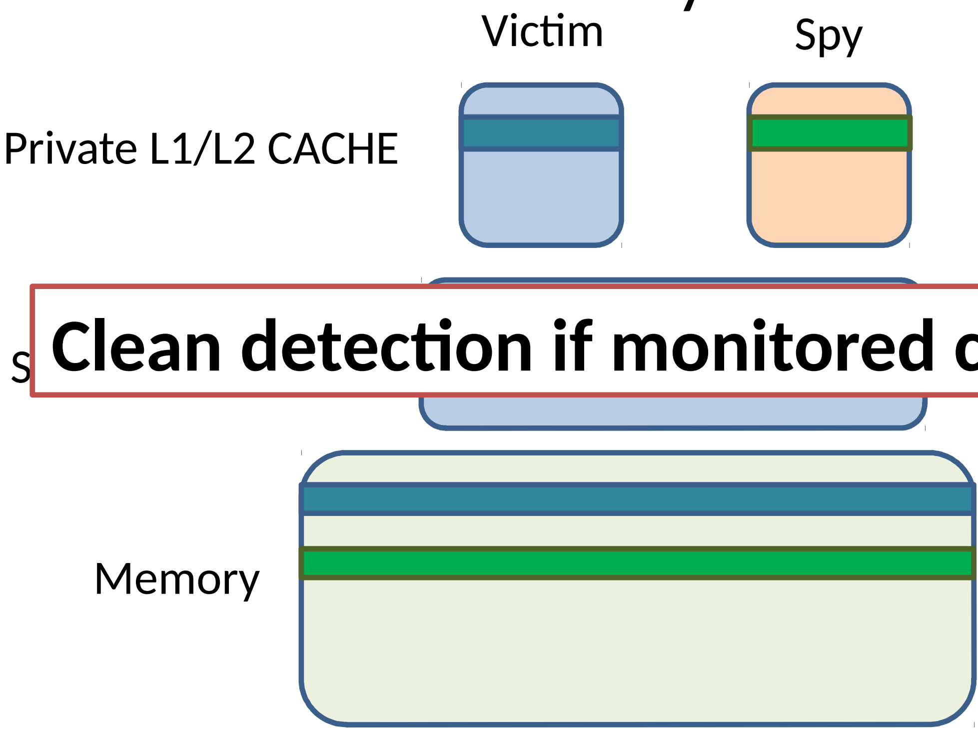
- Cache Attacks are old [Hu92]
- Popular Method: *Prime+Probe* [OST06]:
  1. **Prime** memory lines
     fill monitored cache set with dummy data: eviction set
  2. Wait for some time
  3. **Probe** memory lines
     read eviction set data and time read
- Difficult in L3-cache due to virtual addressing:
  - **Solution**: **Huge Pages** give control of L3$ to spy: e.g. El Gamal [LY+15] or AES [IES15]

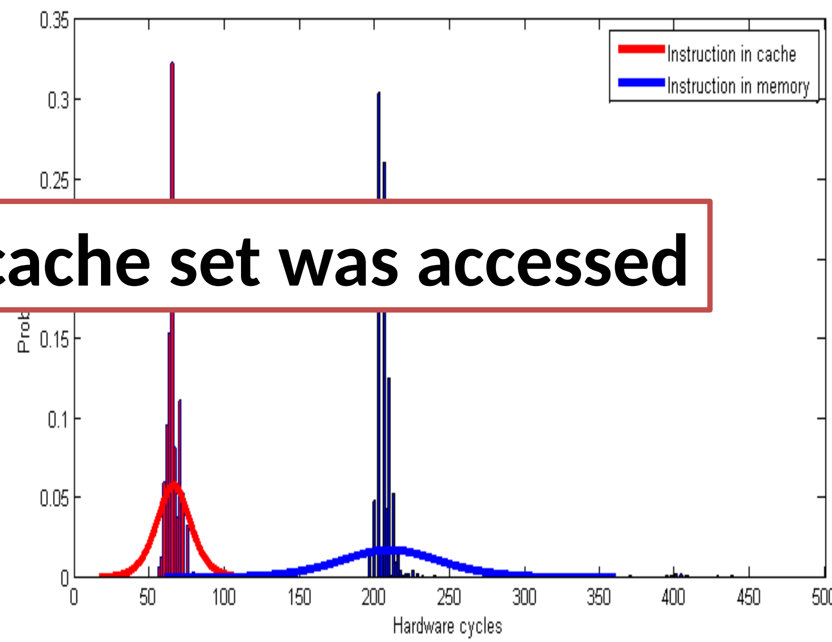
[Hu92] Hu, W.-M. (Digital Equipment Corp., Littleton, MA, USA) *Lattice scheduling and covert channels.* IEEE Oakland 92
[OST06] DA Osvik, A Shamir, E Tromer *Cache attacks and countermeasures: the case of AES*. CT-RSA 2006
[LY+15] Liu, F., Yarom, Y., Ge, Q., Heiser, G., & Lee, R. B. (2015). Last-Level Cache Side-Channel Attacks are Practical. (S&P 2015).
[IES15] Irazoqui, G., Eisenbarth, T., & Sunar, B. S$A: A shared cache attack that works across cores and defies VM sandboxing—and Its application to AES. 36th IEEE Symposium on Security and Privacy (S&P 2015)

Prime+Probe Attack: Concept

Steps: (Preparation: Find **eviction set**)

1. **Prime** desired memory lines
2. Wait for some time
3. **Probe** memory lines and measure reload time.





Clean detection if monitored cache set was accessed

How to get Crypto keys?
Modular Exponentiation for RSA

Basic principle: Scan exponent bits from left to right and square/multiply operand accordingly → **Exponent is secret key**

Algorithm: Square-and-Multiply

Input: Exponent H, base element x, Modulus N

Output: $y = x^H \mod N$

1. Determine binary representation $H = (h_t, h_{t-1}, \ldots, h_0)_2$
2. **FOR** $i = t-1$ **TO** 0
3. $y = y^2 \mod N$
4. **IF** $h_i = 1$ **THEN**
5. $y = y * x \mod N$
6. **RETURN** y

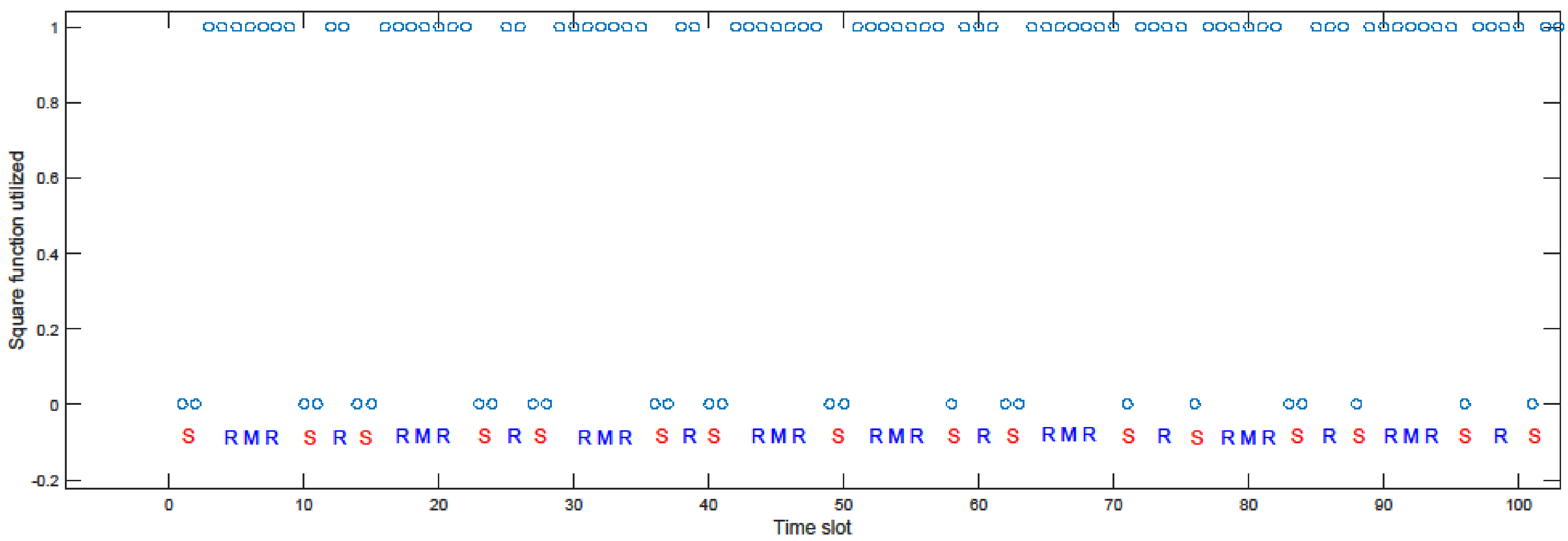
Execution of multiply depends on secret

How to get crypto keys?

Detect **key-dependent** cache accesses:

- **RSA/ElGamal:** Square and Multiply Exponentiation

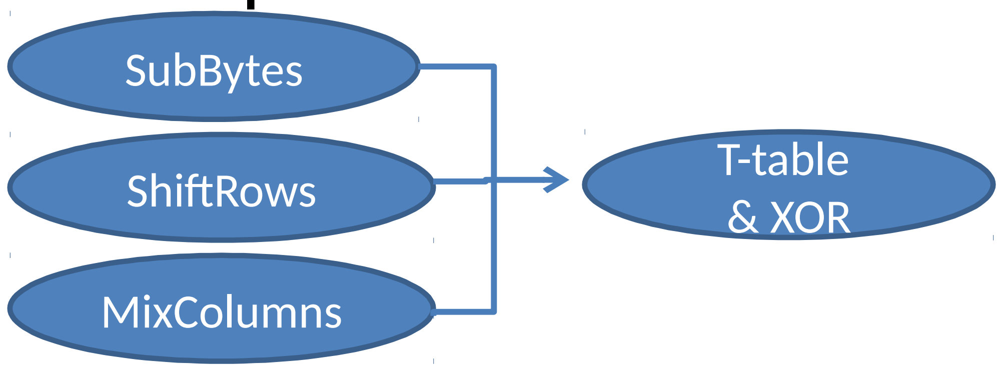
Occurrence of Square (or MUL) in cache reveals key

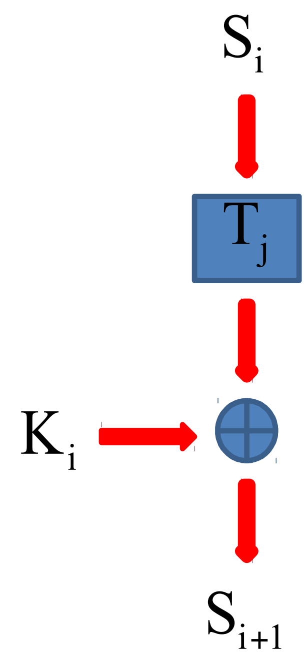


[YF14] Y Yarom, KE Falkner *Flush+ Reload: a High Resolution, Low Noise, L3 Cache Side-Channel Attack,* USENIX Security 2014

Target Cipher: AES

AES T-table implementation:



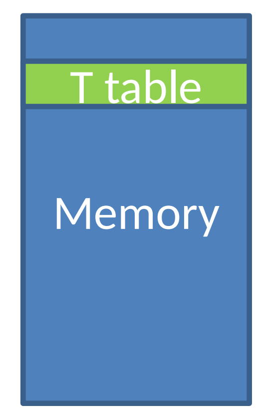


- T-tables stored in memory/cache

Idea:

Detect T-table accesses in last round

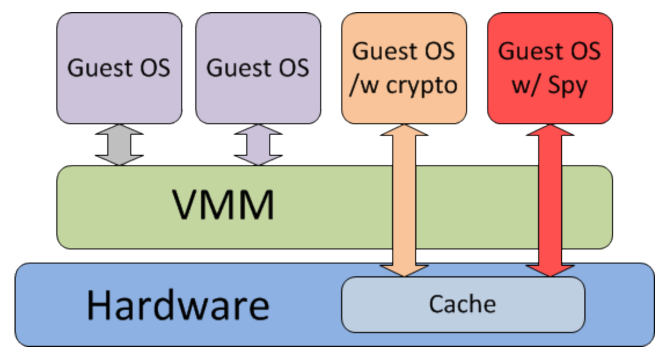
Inclusive caches ensure T-table in LLC



Cloud Cache Attacks

Cache Attacks on Cloud Computing?

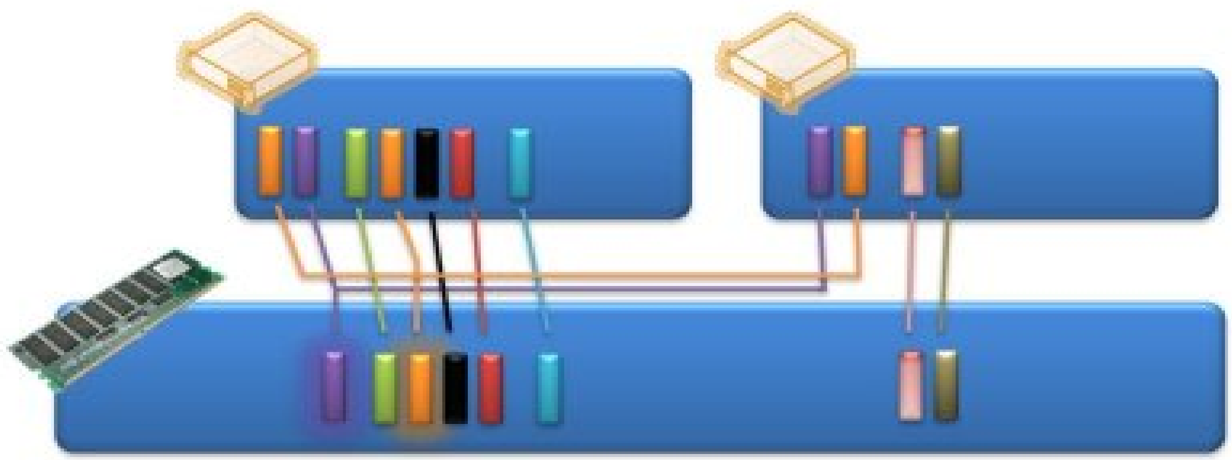
- CSPs: many users on shared, homogeneous platforms
- **Shared resources → Information Leakage?**
  - Adversary and victim share full access to L3 cache
  - Cross Core: L3 Cache is unified cross-core resource



How to track victim’s data? **Shared Memory**

- Sy
- De
  - –
  - → '

OSs)



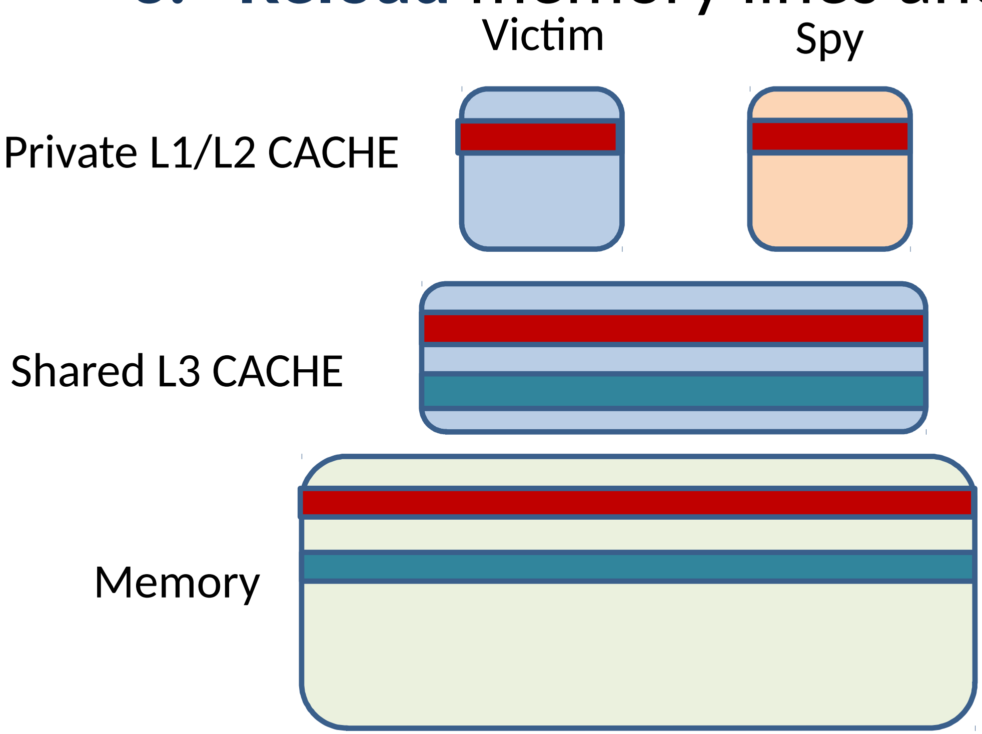
With TPS,duplicate pages are stored once

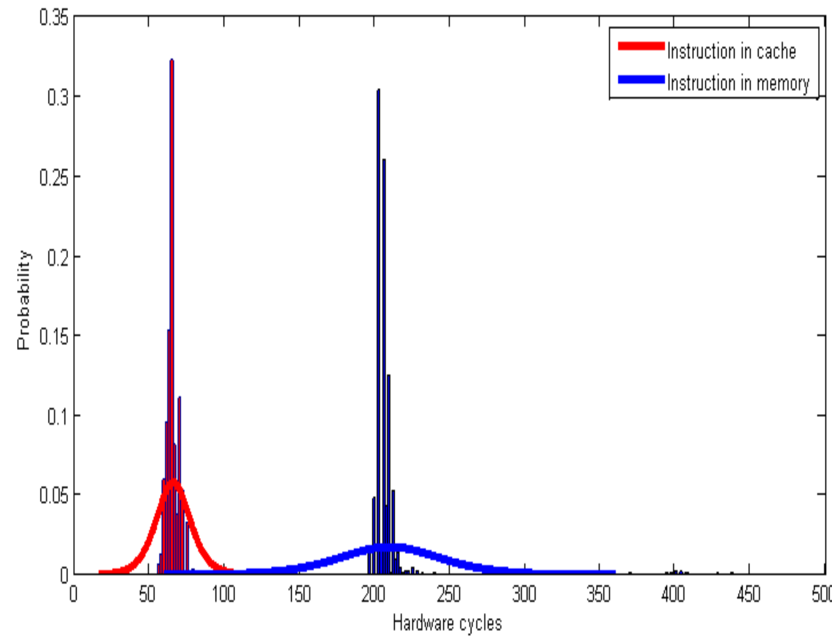
- When Target VM accesses page
  - page copied to cache: copy in shared LLC
  - Subsequent Spy VM access also faster!
  - → Spy can detect Target VMs accesses to known pages

Flush+Reload Attack: Concept

Steps:

1. **Flush** desired memory lines
2. Wait for some time
3. **Reload** memory lines and measure reload time.



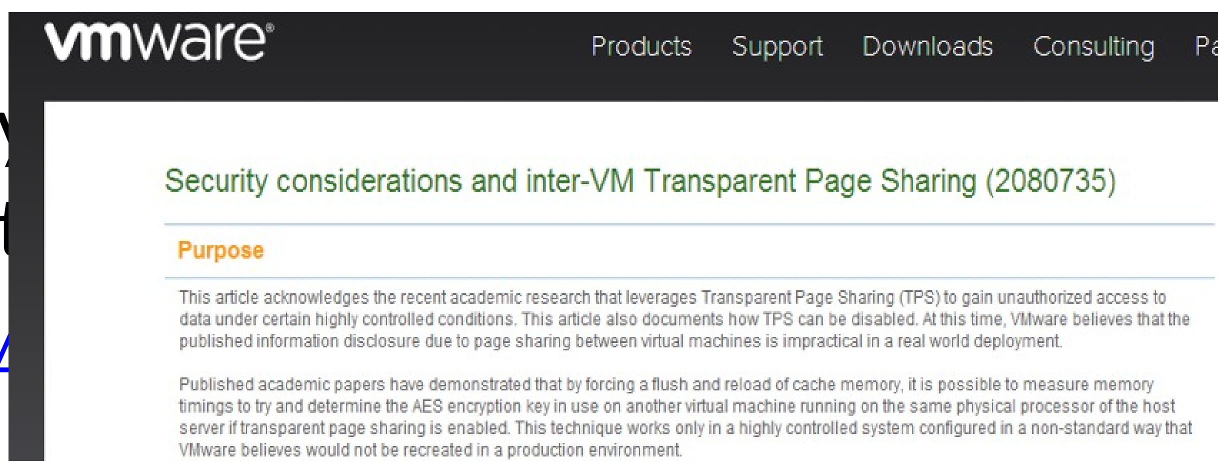


Are Cross-VM Cache Attacks Realistic?

Cross-VM Flush+Reload Attacks work if

- Server has a shared level of cache
- Attacker and the victim are physically co-located
- VMM implements memory deduplication

- Memory ... M cache at
  - http://


vmware®

Products Support Downloads Consulting Par

Security considerations and inter-VM Transparent Page Sharing (2080735)

Purpose

This article acknowledges the recent academic research that leverages Transparent Page Sharing (TPS) to gain unauthorized access to data under certain highly controlled conditions. This article also documents how TPS can be disabled. At this time, VMware believes that the published information disclosure due to page sharing between virtual machines is impractical in a real world deployment.

Published academic papers have demonstrated that by forcing a flush and reload of cache memory, it is possible to measure memory timings to try and determine the AES encryption key in use on another virtual machine running on the same physical processor of the host server if transparent page sharing is enabled. This technique works only in a highly controlled system configured in a non-standard way that VMware believes would not be recreated in a production environment.

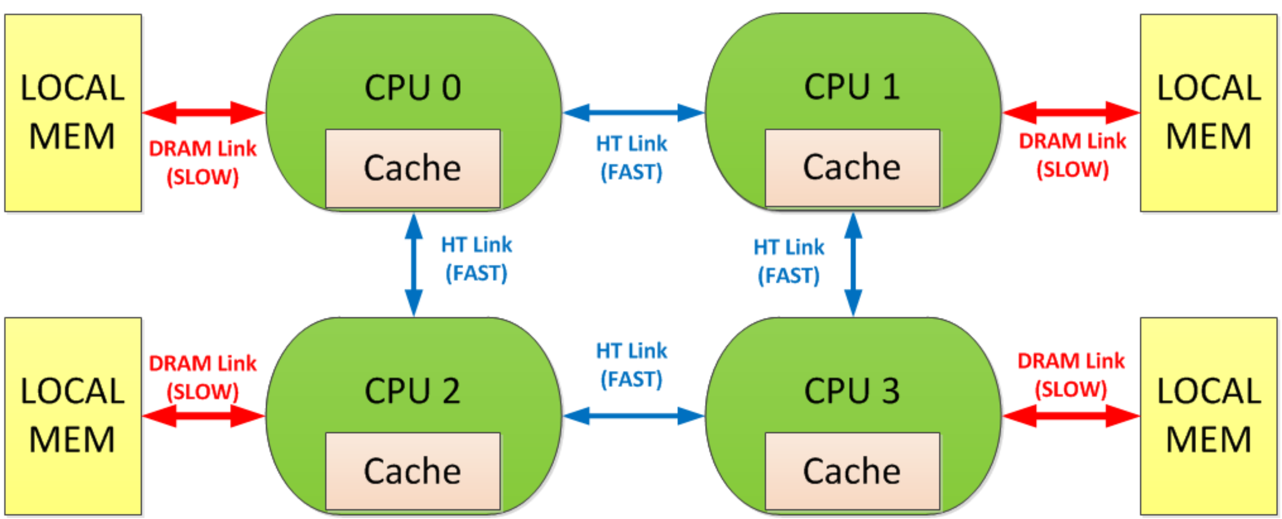
First successful Cache-Attack in Amazon IaaS Cloud

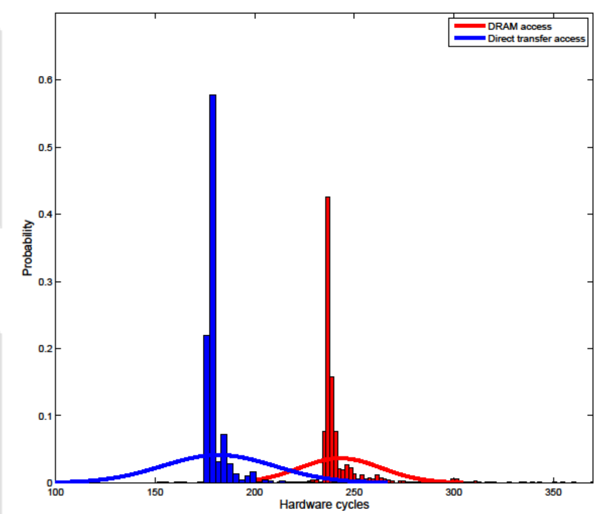
- Full RSA key recovery on EC2:
  - Using Prime & Probe, since it works
  - Co-location via LLC channel
- Major Crypto Libraries (openSSL/Libgcrypt) are widely patched
- Most users in cloud use outdated libraries
  - Targets of opportunity instead of targeted attac
- How to protect non-cryptographic Code?

[IGI+16] M. S. Inci, B. Gulmezoglu, G. Irazoqui, T. Eisenbarth, and B. Sunar: *Cache Attacks Enable Bulk Key Recovery on the Cloud*, CHES 2016

Cross Processor Cache Attacks?

- Cross Processor Data Transfer:
  Cache Coherence Protocols use direct links
  → faster response and less memory B/W
- Faster Accesses→ Data-dependent access time!

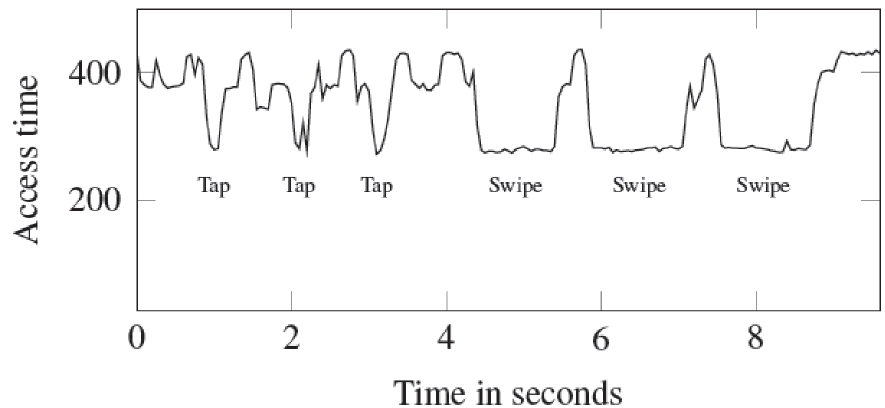




[IES15] G Irazoqui and T Eisenbarth and B Sunar *Cross Processor Cache Attacks* AsiaCCS 2016

Cache Attacks on ARM

- First Attacks: timing attacks (low resolution)
- **ARMageddon**[LGS+16]: First successful Hi-Res Attack
  - Clever cache access strategies to handle replacement policies → essential for success
  - Finds alternative timers and Evict strategies
  - Demonstrates Prime+Probe and Flush/Evict+Reload attacks
    - Key strokes
    - AES T-Tables
    - TrustZone
- ARM Performance feature makes Prime&Probe slightly harder [GRZ+17]

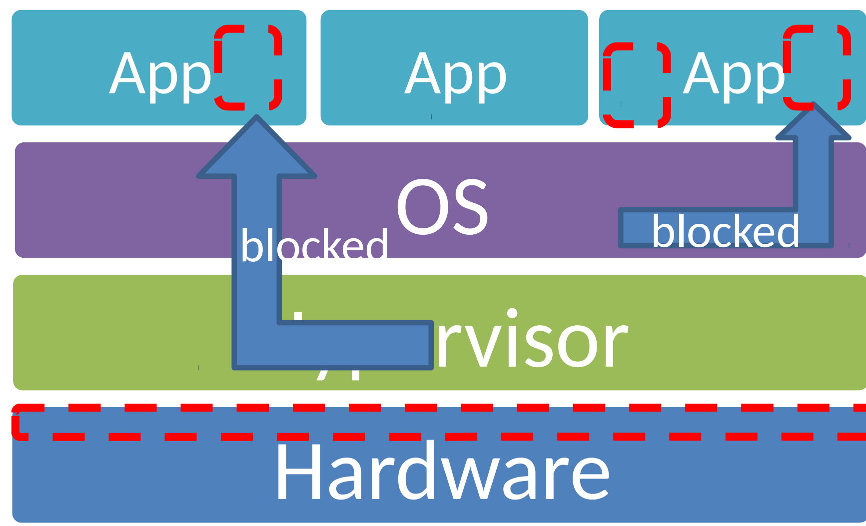


[LGS+16] M. Lipp, D. Gruss, R. Spreitzer, C. Maurice, and S. Mangard: *ARMageddon: Cache Attacks on Mobile Devices* USENIX Security 2016
[GRZ+17] M. Green, L. Rodrigues-Lima, A. Zankl, G. Irazoqui, J. Heyszl, T. Eisenbarth *AutoLock: Why Cache Attacks on ARM Are Harder Than You Think*. USENIX Security 2017

Cache Attacks on Intel SGX

Intel Software Guard Extensions (SGX)

- Trusted Execution Environment
- **Enclave:** Hardware protected user-level software module
  – Loaded by the user program
  – Mapped by the Operating System
  – Authenticated and Encrypted by CPU
- Protects against system level adversary
- "no protection against access pattern leakages"



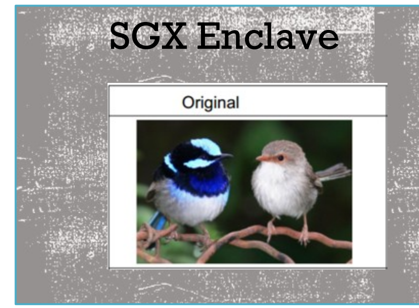
New Attacker Model:

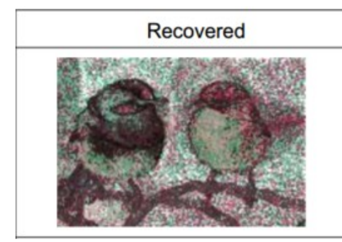
Attacker gets full control over OS

Side Channel Attacks on SGX

OS initiated attacks are powerful:

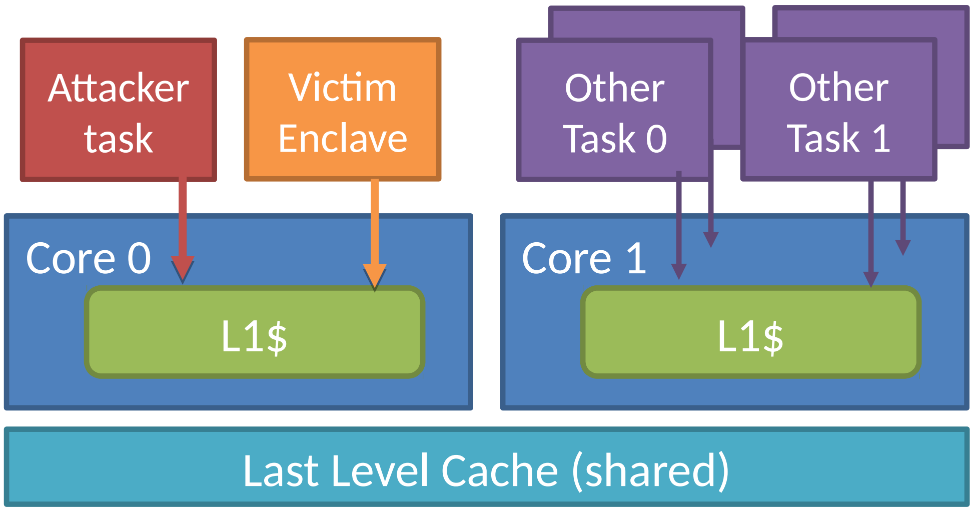
- Page Accesses [XCP15, vBWK+17]
- Branch Shadowing [LSG+17]
- Cache Attacks
  - Classic [GESM17, BMD+17]
  - Enclave to Enclave [SWG+17]





[XCP15] Yuanzhong Xu, Weidong Cui, Marcus Peinado. *Controlled-channel attacks: Deterministic side channels for untrusted operating systems*. IEEE S&P, 2015.
[vBWK+17] J. Van Bulck, N. Weichbrodt, R. Kapitza et al. *Telling Your Secrets without Page Faults: Stealthy Page Table-Based Attacks on Enclaved Execution.* Usenix Security 17.
[LSG+17] Sangho Lee, Ming-Wei Shih, Prasun Gera, et al. *Inferring Fine-grained Control Flow Inside SGX Enclaves with Branch Shadowing*. Usenix Security 17.
[GESM17] Götzfried, J., Eckert, M., Schinzel, S., Müller, T.: *Cache Attacks on Intel SGX*. EUROSEC 17
[BMD+17] Ferdinand Brasser,, Urs Müller, Alexandra Dmitrienko et al. *Software Grand Exposure: SGX Cache Attacks Are Practical*. WOOT 17
[SWG+17] Schwarz, M., Weiser, S., Gruss, D., Maurice, C., Mangard, S: *Malware guard extension: Using SGX to conceal cache attacks*. DIMVA 2017

CacheZoom:
High Resolution Cache Attack on SGX



Full control over OS:

- Prime+Probe Attack
- Isolate Cores: Same-Core L1C Monitoring
- CPU Freq. fixed
- **Interrupted Execution:** Full Cache image every few instructions

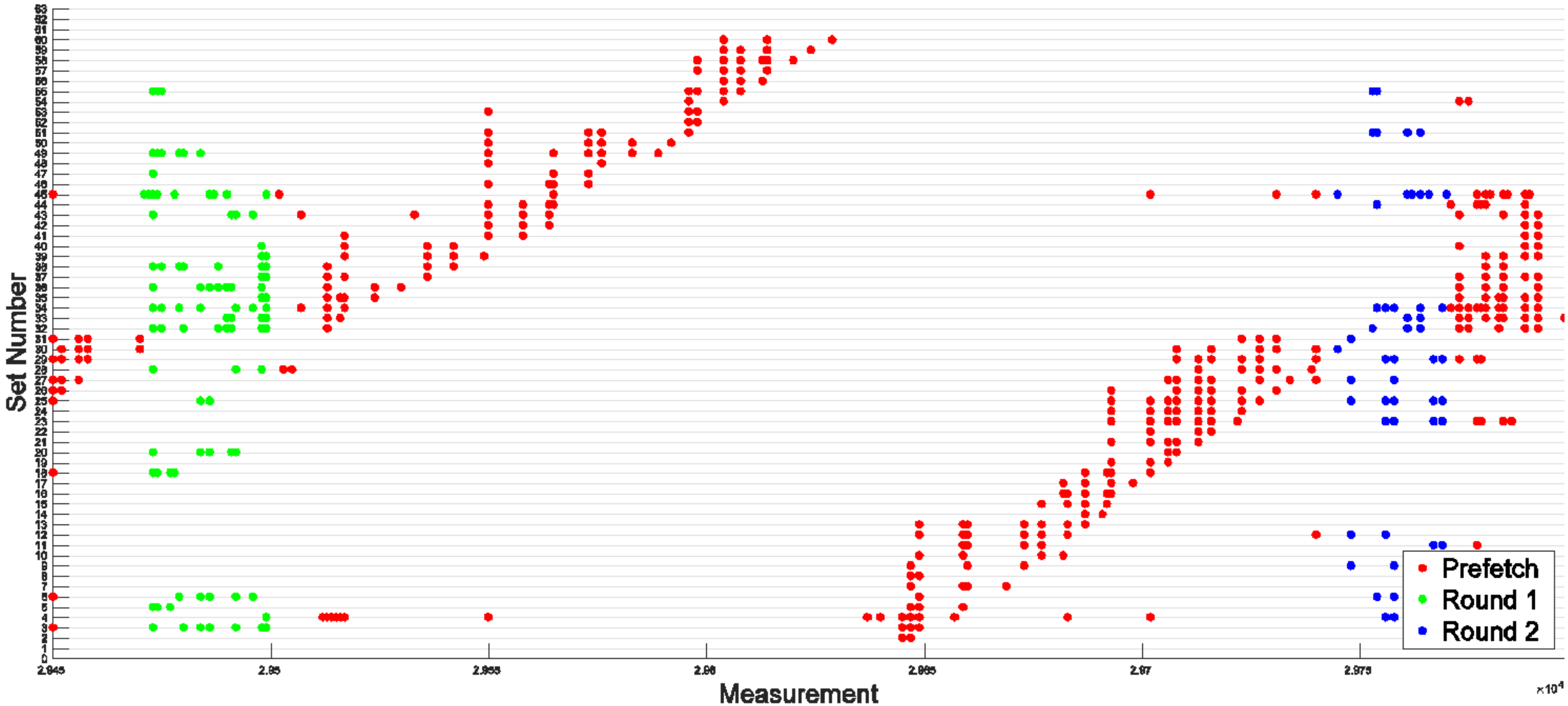
Sample Target: AES

➢ **All** table-based implementations vulnerable

➢ Even Cache-warming (table prefetch) ineffective

[MIE17] Moghimi, A. , Irazoqui, G., Eisenbarth, *CacheZoom: How SGX Amplifies The Power of Cache Attacks* CHES 2017

CacheZoom: AES Trace



Meltdown & Spectre

Cache Speculation Side Channels

Speculative Execution

- Loads data without security checks
- Rolls back state before committing
- Cache state influenced, **but never rolled back!**

Idea: 1. read privileged info

2. leak via cache *access pattern*

MeltDown:
Exploiting Out-of-Order Execution

Uses out-of-order execution to leak kernel space memory

- Exceptions prevent access to kernel space (supervisor bit set on kernel page)
- Exceptions checked before *commit*
  ➔ *after* data is read/spec. processed

Idea: use out-of-order execution to leak privileged data *before* exception check

1. Read bit from Kernel Space
2. Access [address + bit<<6]

MeltDown: Reading Privileged Memory

Process 1:

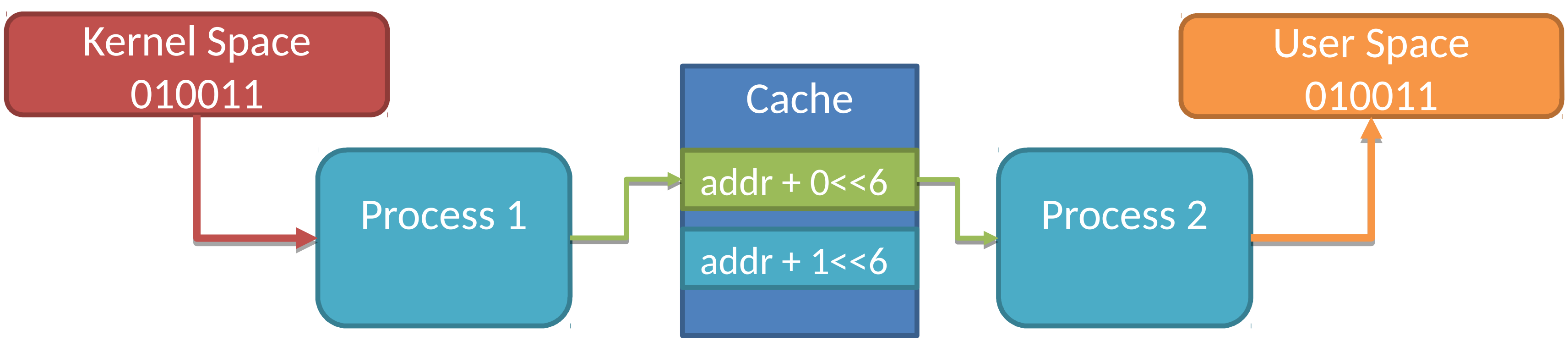
Read and leak sensitive data

1. Read sensitive bit
2. Access [addr + bit]
3. (recover from exception)

Process 2:

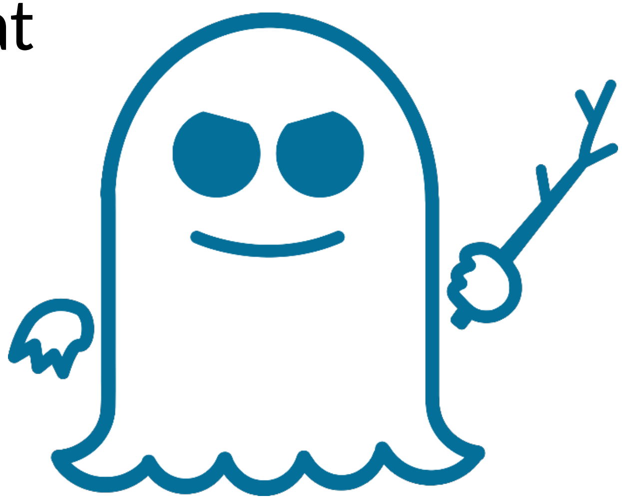
Read and store leakage

1. Flush [addr + x]
2. Wait
3. Reload [addr + x]
4. (write out result)



SPECTRE: Speculative Execution Attack

- Tricks victim code to leak sensitive data in its memory space
- Victim code contains code gadget that
  - Reads sensitive data speculatively
  - leaks data through execution trace
- Attacker activates gadget
  - Either through poisoned input
  - Or by creating new false (speculative) execution path through training BTB
- Attacker reads data from cache trace



MeltDown / Spectre: Summary

- First time *register contents* are leaked by microarchitectural attack
- Meltdown mostly fixed
  - switch to kernel mode becomes slow
- Spectre: not clear, fences help, but can be avoided? → Exploit base for years to come?
- CERT recommends:

Solution

Replace CPU hardware

The underlying vulnerability is primarily caused by CPU architecture design choice requires replacing vulnerable CPU hardware.

Preventing
Cache Attacks

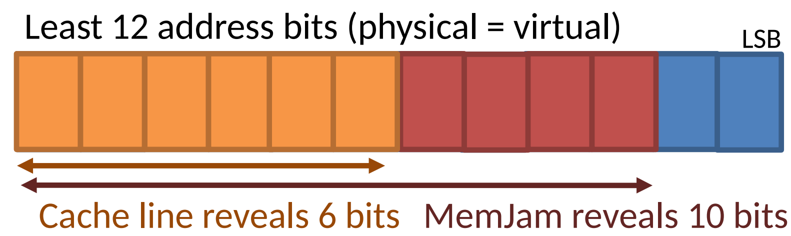
Cache Attack Prevention

Write unexploitable Code

- Constant execution time
- Secret-independent execution flow
- Secret-independent memory accesses

Intra Cache Line Leakages

- **Idea:** Cache attackers get *cache line* granularity (64 byte on Intel)



- Used in some “constant-time” implementations and in code verification tools

Counterexamples:

- **CacheBleed** [YGH16]: Exploits L1C Banking
  (not in 6th and 7th Gen Intel→ not applicable to SGX)
- **MemJam**[MES18]: Exploits False Dependency Checks
  works in all modern Intel CPUs→applicable to SGX

[YGH16] Y. Yarom, D. Genkin, and N. Heninger: *CacheBleed: A Timing Attack on OpenSSL Constant Time RSA*, CHES 2016 and JCEN 2017
[MES17] Moghimi, A., Eisenbarth, T. and Sunar, B., *MemJam: A False Dependency Attack against Constant Time Crypto Implementations in SGX*; accepted at CT-RSA 2018 https://arxiv.org/abs/1711.08002

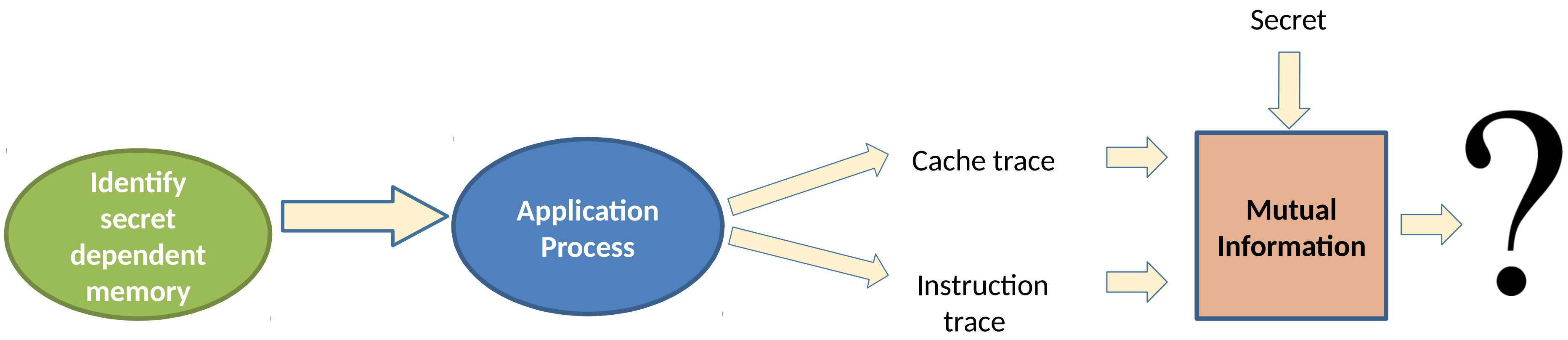
Detecting Vulnerable Code

- Static Analysis
  - CacheAudit [DKMR15]
- Dynamic Analysis
  - LLVM Level [ABB+16]
  - Symbolic Execution [WWP+17]
  - PIN Trace [ZHS17]
  - Actual execution on machine [IGK+17]

[DKMR15] Doychev, G., Köpf, B., Mauborgne, L. and Reineke, J.: *Cacheaudit: A tool for the static analysis of cache side channels*. *ACM TISSEC*, *18*(1), 2015
[ABB+16] Almeida, J.B., Barbosa, M., Barthe, G., Dupressoir, F. and Emmi, M. *Verifying Constant-Time Implementations*. USENIX Security 2016
[WWP+17] Wang, S., Wang, P., Liu, X., Zhang, D. and Wu, D., *CacheD: Identifying Cache-Based Timing Channels in Production Software*. USENIX Security 2017
[ZHS17] A. Zankl, J.Heyszl, and G. Sigl.: *Automated Detection of Instruction Cache Leaks in RSA Software Implementations*. In CARDIS 2016
[IGK+17] G. Irazoqui, X. Guo, H. Khattri, A. Kanuparthi, T. Eisenbarth, B. Sunar: *Did we learn from LLC Side Channel Attacks? A Cache Leakage Detection Tool for Crypto Libraries* arXiv: https://arxiv.org/abs/1709.01552

Cache Leakage Free Code Verification

- Ensure there are no secret dependent branches/memory accesses in *final code*
- **Our approach:**
  1. Detect secret dependent branches/accesses through *taint analysis*
  2. Obtain cache traces of those instructions/variables
  3. Check for Mutual Information with sensitive values



Finding leakages in Cryptographic Code

Analyzed RSA, ECC and AES of major crypto libraries:

- 50% of the implementations leaked information (2016)
- We notified and help fixing these vulnerabilities
  - WolfSSL
    - CVE 2016-7438,7439,7440
  - Intel IPP
    - CVE 2016-8100
  - Bouncy Castle
    - CVE 2016-10003323

Cryptographic Primitive	Library	Outcome
AES	OpenSSL (T-table)	Leaks
	OpenSSL (S-box)	No leak
	WolfSSL	Leaks
	IPP (v1)[1]	No leak
	IPP (v2)[1]	No leak
	LibreSSL (S-box)	No leak
	NSS	Leaks
	Libgcrypt	No leak
	Bouncy Castle	Leaks
	MbedTLS	Leaks
RSA	OpenSSL (Sliding W)	Leaks
	OpenSSL (Fixed W)	No leak
	WolfSSL (Montgomery L)	Leaks
	WolfSSL (Sliding W)	Leaks
	IPP	Leaks
	LibreSSL	No leak
	NSS	No leak
	Libgcrypt	No leak
	Bouncy Castle (Sliding W)	Leaks
	MbedTLS (Sliding W)	Leaks
ECC	**OpenSSL (WNAF)**[2]	Leaks
	WolfSSL (Montgomery L)	Leaks
	WolfSSL (Sliding W)	Leaks
	IPP[1]	No leak
	LibreSSL	Leaks
	NSS	No leak
	Libgcrypt	No leak
	Bouncy Castle (Fixed W)	Leaks
	MbedTLS (Fixed W)	No leak

[IGK+17] G. Irazoqui, X. Guo, H. Khattri, A. Kanuparthi, T. Eisenbarth, B. Sunar: *Did we learn from LLC Side Channel Attacks? A Cache Leakage Detection Tool for Crypto Libraries* arXiv: https://arxiv.org/abs/1709.01552

Conclusion

- Cache Attacks are powerful
  - Very effective on TEEs such as SGX with OS control
  - Still fully functional in Cloud and standalone systems
  - A great tool to spread speculative results
- Constant time code still best defense
  - But no longer sufficient, thanks to SPECTRE

Thank You!

vernam.wpi.edu

its.uni-luebeck.de

thomas.eisenbarth@uni-luebeck.de