black hat ASIA 2017

Cache Side Channel Attack: Exploitability and Countermeasures

Gorka Irazoqui

Xiaofei (Rex) Guo, Ph.D.

girazoki *noSPAM* wpi.edu

xiaofei.rex.guo *noSPAM* tetrationanalytics.com

Who are We?

- Gorka Irazoqui
  - PhD candidate in WPI
  - Intern at Intel in summer 2016
  - Focus on micro-architectural attacks

Who are We?

- Xiaofei (Rex) Guo
  - Technical lead at Cisco Tetration Analytics
    - Visibility to everything in data center in real time
    - Automated and dynamic policy generation and enforcement
  - Worked at Intel Security Center of Excellence and Qualcomm Product Security Initiative
    - IoT and mobile platform security, infrastructure security, and application security
  - PhD from New York University

Disclaimer

We don’t speak for our employer. All the opinions and information here are our responsibility including mistakes and bad jokes.


1
2
3
4
5
6
7
8
9
*
0
#


2
4
5
7
9
*
0
#

“You must be kidding, cache attacks are not practical!”

RISK ASSESSMENT —

Storing secret crypto keys in the Amazon cloud? New attack can steal them

Technique allows full recovery of 2048-bit RSA key stored in Amazon's EC2 service.

DAN GOODIN - 9/28/2015, 2:55 PM

"You must be kidding, cache attacks are not practical!"

RISK ASSESSMENT —

Storing secret crypto keys in the Amazon cloud? New attack can steal them

Technique allows full recovery of 2048-bit RSA key stored in Amazon's EC2 service.

DAN GOODIN - 9/28/2015, 2:55 PM

"You must be kidding, cache attacks are not practical!"

Security considerations and disallowing inter-Virtual Machine 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 and documents VMware's precautionary measure of restricting TPS to individual virtual machines by default in upcoming ESXi releases. At this time, VMware believes that the published information disclosure due to TPS between virtual machines is impractical in a real world deployment.

RISK ASSESSMENT —

Storing secret crypto keys in the Amazon cloud? New attack can steal them

Technique allows full recovery of 2048-bit RSA key stored in Amazon's EC2 service.

DAN GOODIN - 9/28/2015, 2:55 PM

"You must be kidding, cache attacks are not practical!"

Security considerations and disallowing inter-Virtual Machine 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 and documents VMware's precautionary measure of restricting TPS to individual virtual machines by default in upcoming ESXi releases. At this time, VMware believes that the published information disclosure due to TPS between virtual machines is impractical in a real world deployment.

CacheBleed OpenSSL Vulnerability Affects Intel-Based Cloud Servers

Only Sandy Bridge (and earlier) Intel CPUs are affected

Mar 2, 2016 08:26 GMT · By Catalin Cimpanu · Share:

Yesterday's OpenSSL updates (1.0.2g and 1.0.1s) not only brought a fix against the already infamous DROWN attack but also patched seven other security flaws, one labeled as high, one moderate, and five as low severity.

RISK ASSESSMENT —

Storing secret crypto keys in the Amazon cloud? New attack can steal them

Technique allows full recovery of 2048-bit RSA key stored in Amazon's EC2 service.

DAN GOODIN - 9/28/2015, 2:55 PM

"You must be kidding, cache attacks are not practical!"

Security considerations and disallowing inter-Virtual Machine 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 and documents VMware's precautionary measure of restricting TPS to individual virtual machines by default in upcoming ESXi releases. At this time, VMware believes that the published information disclosure due to TPS between virtual machines is impractical in a real world deployment.

CacheBleed OpenSSL Vulnerability Affects Intel-Based Cloud Servers

Only Sandy Bridge (and earlier) Intel CPUs are affected

Mar 2, 2016 08:26 GMT · By Catalin Cimpanu · Share:

Yesterday's OpenSSL updates (1.0.2g and 1.0.1s) not only brought a fix against the already infamous DROWN attack but also patched seven other security flaws, one labeled as high, one moderate, and five as low severity.

ANDROID DEVICES VULNERABLE TO ARMAGEDDON CACHE ATTACK

SECURITY NEWS | AUGUST 15, 2016 | 0 | BY JOSEPH STEINBERG

The paper ARMageddon: Cache Attacks on Mobile Devices have been included in 25th USENIX Security Symposium. The

Feasibility Trend

Cache Attacks and Countermeasures: The Case of AES

Article *in* Lecture Notes in Computer Science 2005 · January 2005 *with* 26 Reads
DOI: 10.1007/11605805_1 · Source: DBLP

1st Dag Arne Osvik
4.67 · Customs Solutions Group S.A.

2nd Adi Shamir

3rd Eran Tromer
14.96 · Tel Aviv University

Intel, Spark, AMD | Linux | OpenSSL AES

Feasibility Trend

Cache Attacks and Countermeasures: The Case of AES

Yet another MicroArchitectural Attack: : exploiting I-Cache.

Conference Paper (PDF Available) · January 2007 *with* 32 Reads

DOI: 10.1145/1314466.1314469 · Source: DBLP

Conference: Proceedings of the 2007 ACM workshop on Computer Security Architecture, CSAW 2007, Fairfax, VA, USA, November 2, 2007

1st Onur Acıçmez
7.04 · Unknown

Intel, Spark, AMD | Linux | OpenSSL AES

Intel | Linux | OpenSSL RSA

Feasibility Trend

Cache Attacks and Countermeasures: The Case of AES

Yet another MicroArchitectural Attack: : exploiting I-Cache.

FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack

Conference Paper · August 2014 *with* 13 Reads

Conference: USENIX Security Symposium

1st Yuval Yarom
1.71 · University of Adelaide

2nd Katrina Falkner
16.03 · University of Adelaide

Intel, Spark, AMD | Linux | OpenSSL AES

Intel | Linux | OpenSSL RSA

Intel (Cross-core) | Linux (deduplication) | GnuPG RSA

Feasibility Trend

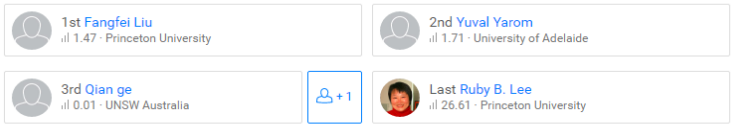
Cache Attacks and Countermeasures: The Case of AES

Yet another MicroArchitectural Attack: : exploiting I-Cache.

FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack

Last-Level Cache Side-Channel Attacks are Practical

Article · July 2015 *with* 30 Reads
DOI: 10.1109/SP.2015.43



Intel, Spark, AMD | Linux | OpenSSL AES

Intel | Linux | OpenSSL RSA

Intel (Cross-core) | Linux (deduplication) | GnuPG RSA

Intel (Cross-Core) | Linux (no deduplication) | GnuPG RSA

Feasibility Trend

Cache Attacks and Countermeasures: The Case of AES

Yet another MicroArchitectural Attack: : exploiting I-Cache.

FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack

Last-Level Cache Side-Channel Attacks are Practical

Cross Processor Cache Attacks

Conference Paper · January 2016 *with* 3 Reads

DOI: 10.1145/2897845.2897867

Conference: the 11th ACM

1st Gorka Irazoqui

2nd Thomas Eisenbarth

3rd Berk Sunar

19.39 · Worcester Polytechnic Institute

Intel, Spark, AMD | Linux | OpenSSL AES

Intel | Linux | OpenSSL RSA

Intel (Cross-core) | Linux (deduplication) | GnuPG RSA

Intel (Cross-Core) | Linux (no deduplication) | GnuPG RSA

AMD (cross CPU) | Linux | OpenSSL AES and GnuPG El Gamal

Feasibility Trend

Cache Attacks and Countermeasures: The Case of AES

Yet another MicroArchitectural Attack: : exploiting I-Cache.

FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack

Last-Level Cache Side-Channel Attacks are Practical

Cross Processor Cache Attacks

ARMageddon: Last-Level Cache Attacks on Mobile Devices

Article · November 2015 *with* 34 Reads

Source: arXiv

1st Moritz Lipp

2nd Daniel Gruss

3rd Raphael Spreitzer
2.25 · Graz University of Technology

4th Stefan Mangard

Intel, Spark, AMD | Linux | OpenSSL AES

Intel | Linux | OpenSSL RSA

Intel (Cross-core) | Linux (deduplication) | GnuPG RSA

Intel (Cross-Core) | Linux (no deduplication) | GnuPG RSA

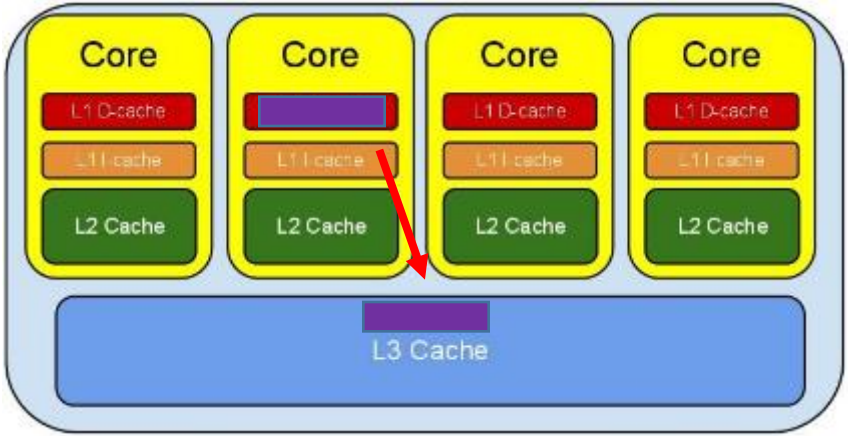
AMD (cross CPU) | Linux | OpenSSL AES and GnuPG El Gamal

ARM (cross core/CPU) | Android | Bouncy Castle AES

Functionality

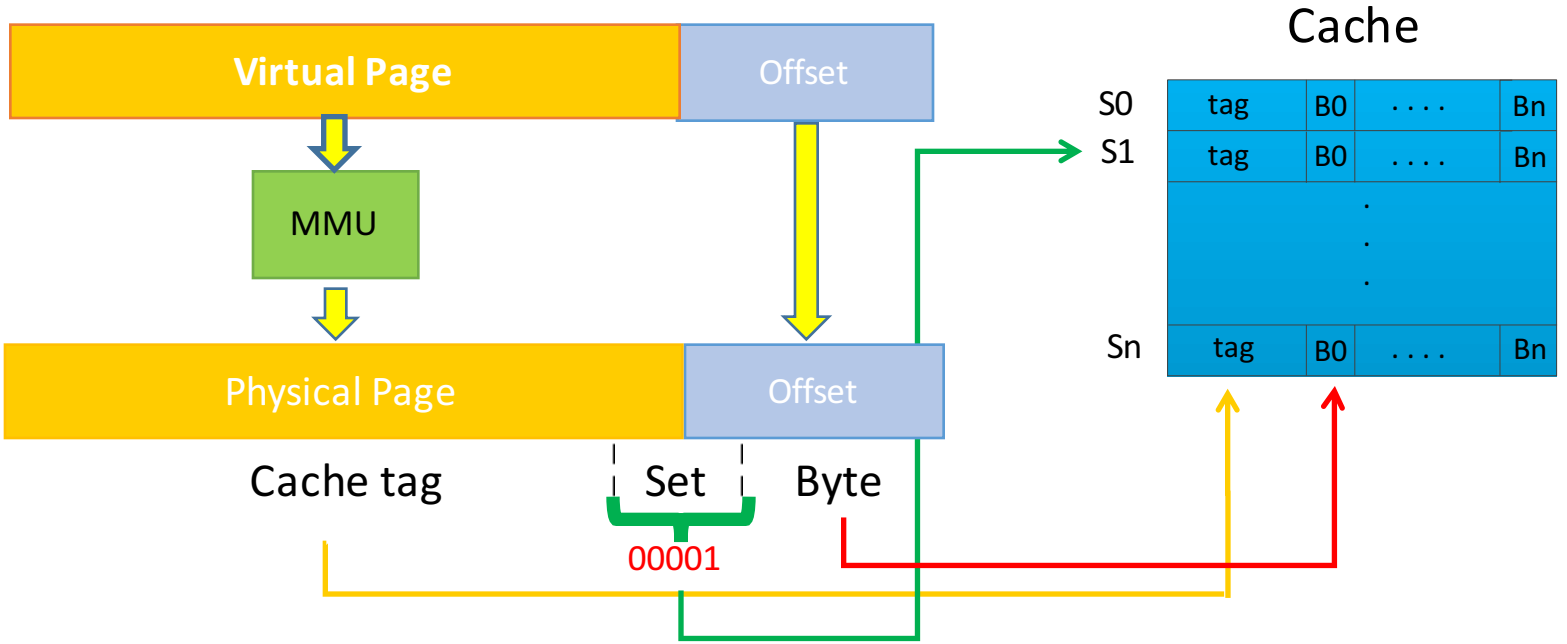
LLC as a Side Channel?

- Caches: fast access memories
- Why would an attacker use LLC as covert channel?
  - Cross-core
  - Inclusiveness
  - High resolution



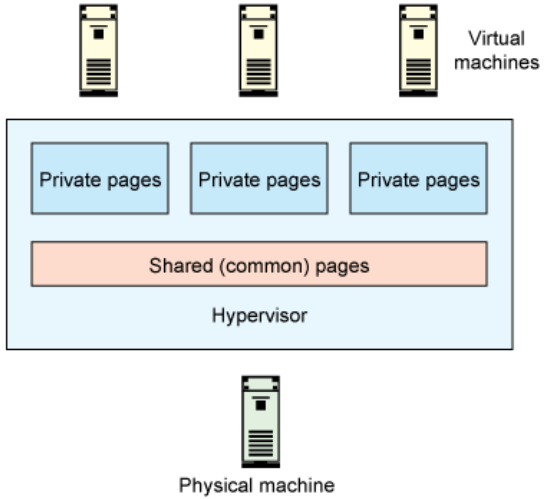
Cache Architecture

- Set associative: cache divided in n-way sets
- Location in the cache determined by physical address



Flush + Reload Attack

- Requirement 1: deduplication
  - Identical read-only memory pages are shared
  - Attacker and victim access the same address
  - Linux and KVM (KSM), Vmware (TPS) and Android (Zygote)
- Requirement 2: flush instruction (e.g., clflush in x86)
- CVE 2014-3356: Vmware enabled deduplication by default



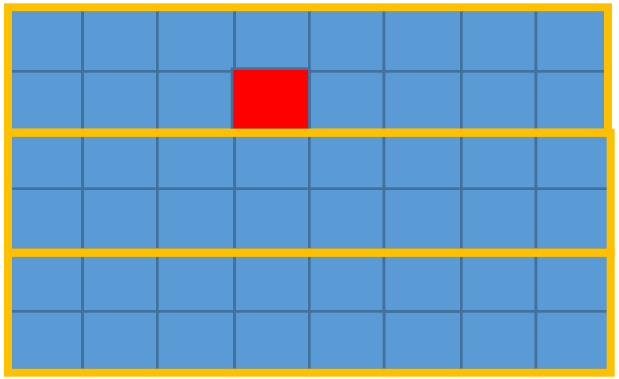
Security considerations and disallowing inter-Virtual Machine Transparent Page Sharing (2080735)

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 and documents VMware's precautionary measure of restricting TPS to individual virtual machines by default in upcoming ESXi releases. At this time, VMware believes that the published information disclosure due to TPS between virtual machines is impractical in a real world deployment.

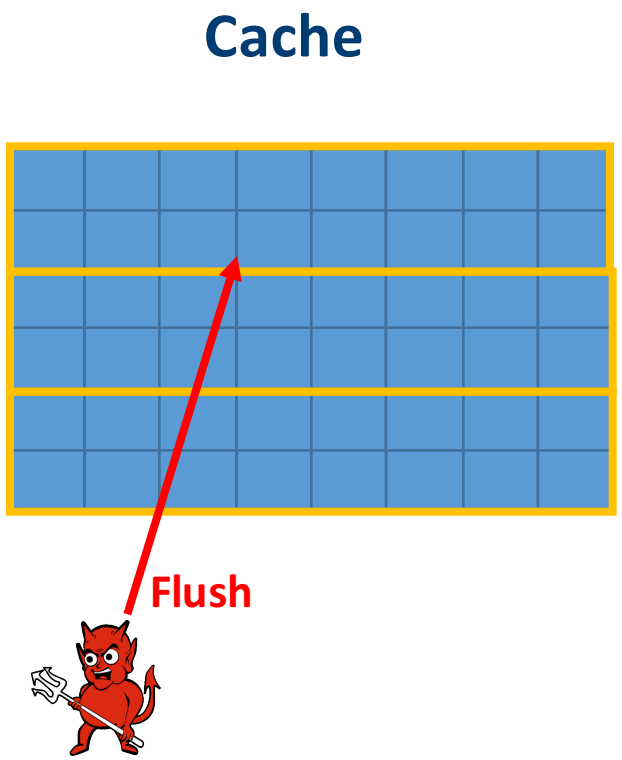
Flush + Reload Attack

- Attacker flushes a cached memory location



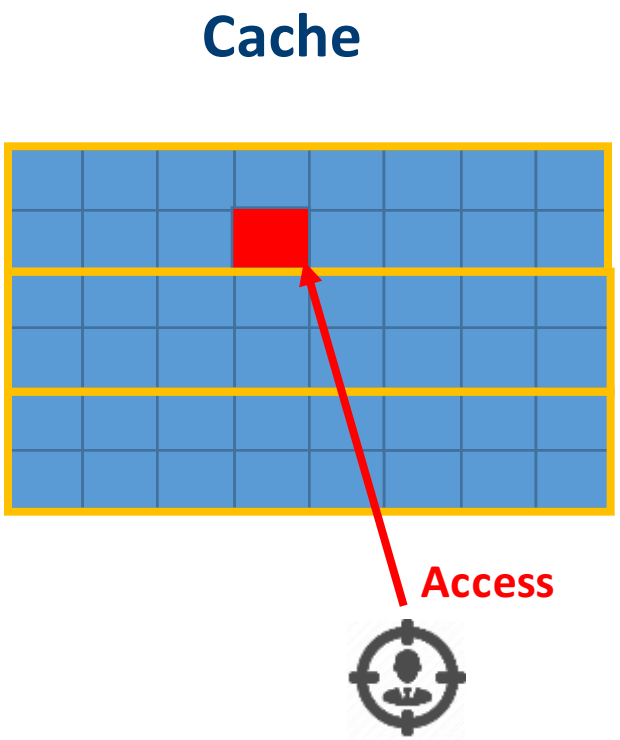
Flush + Reload Attack

- Attacker flushes a cached memory location



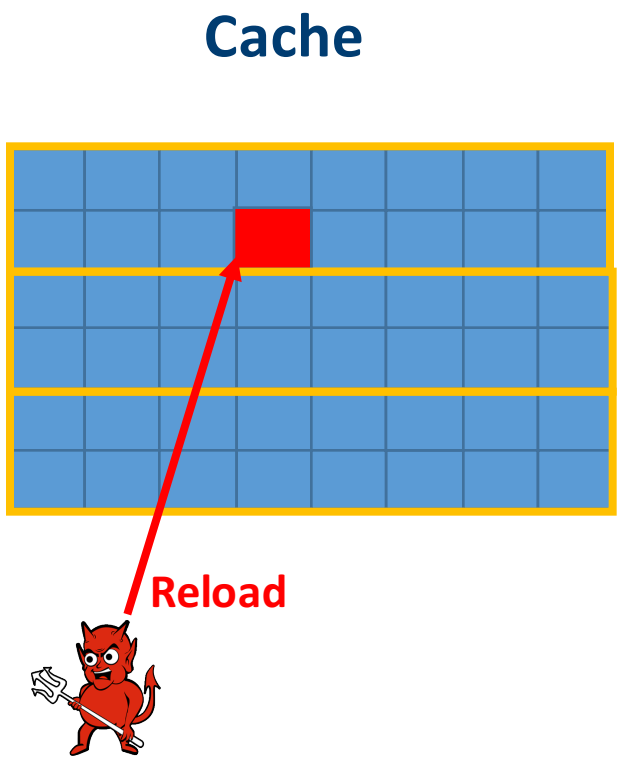
Flush + Reload Attack

- Attacker flushes a cached memory location
- Victim accesses/does not access



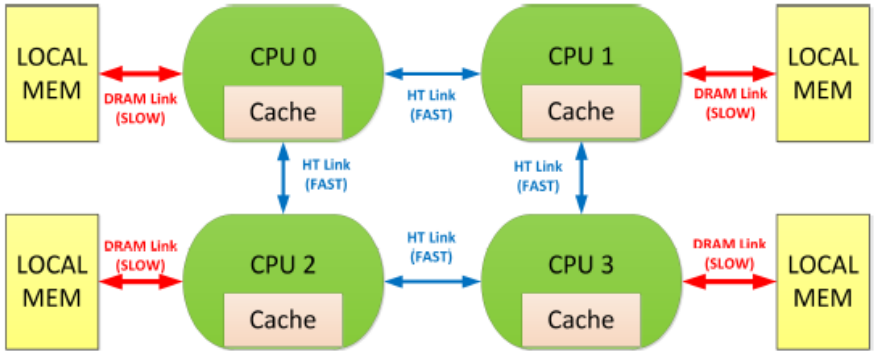
Flush + Reload Attack

- Attacker flushes a cached memory location
- Victim accesses/does not access
- Attacker re-accesses memory location
  - Fast access time -> victim accessed
  - Slow access time -> victim did not access



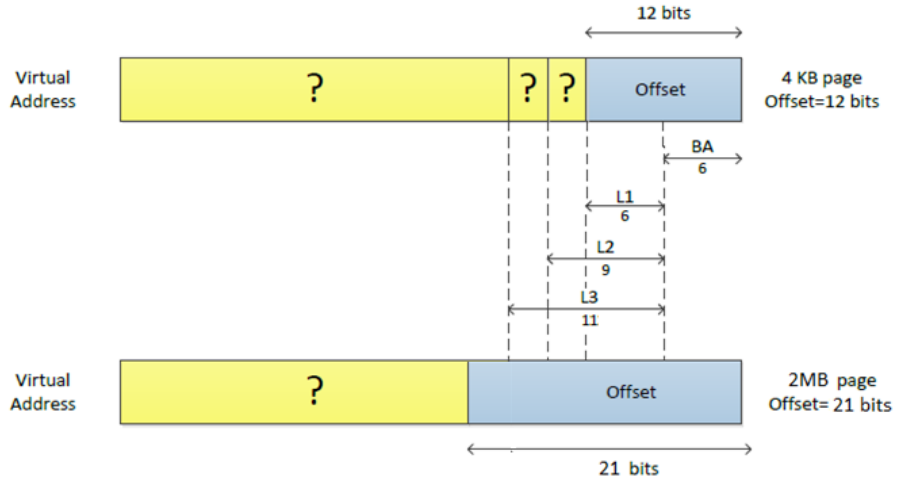
Flush + Reload Attack Summary

- Pros:
  - Low noise: focus on one line, noisy process needs to fill an entire set
  - Applicable across CPU sockets! Flush instruction invalidates memory in other CPUs
  - Works in non-inclusive caches
- Cons:
  - Requirement might be met in some scenarios
  - Can only recover statically allocated data



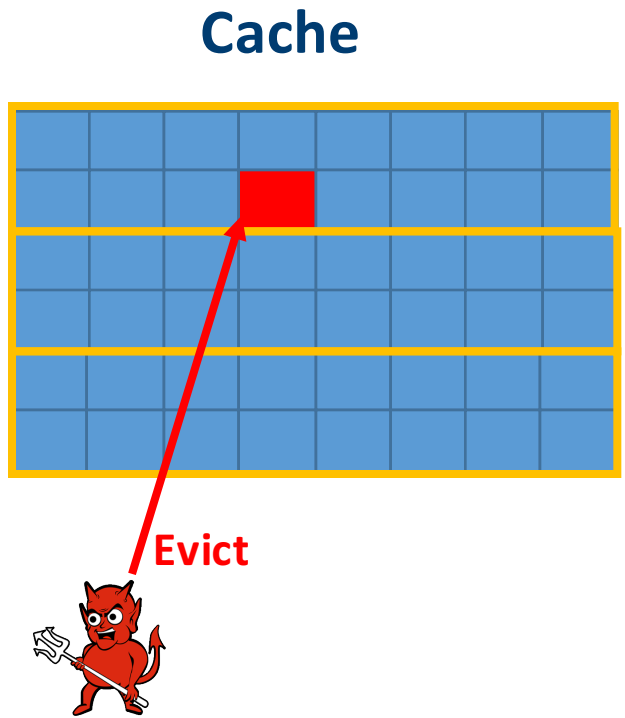
Evict + Reload Attack

- No flush instruction?
- Attacker needs to evict data from LLC
- Attacker can use huge pages
- Physical address selects the set to occupy



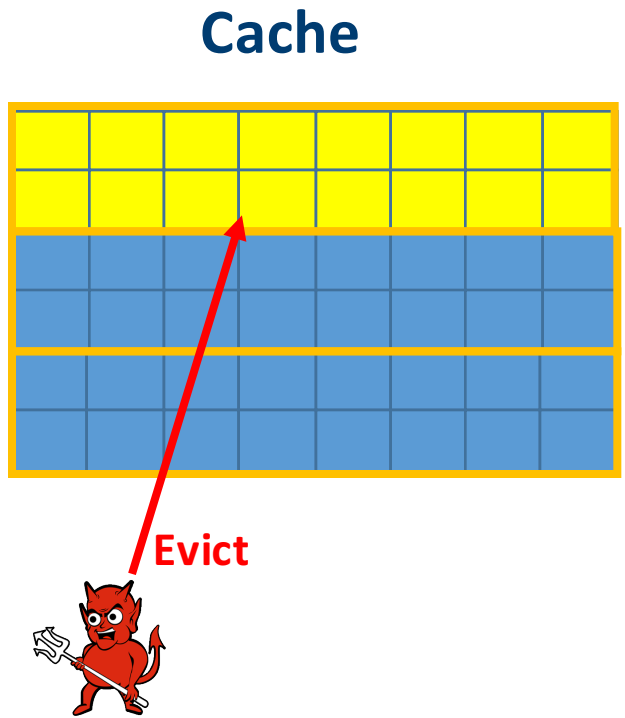
Evict + Reload Attack

- No flush instruction?
- Attacker needs to evict data from LLC
- Attacker can use huge pages
- Physical address selects the set to occupy
- Attacker evicts (fills set)



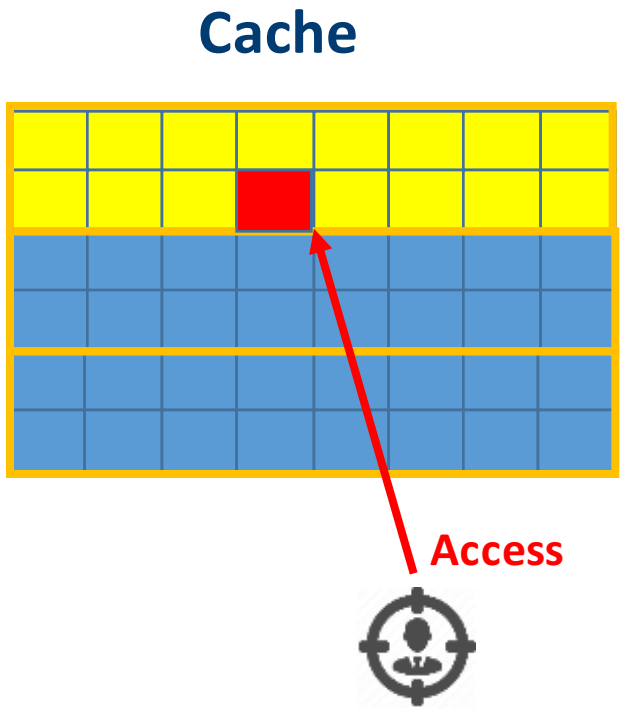
Evict + Reload Attack

- No flush instruction?
- Attacker needs to evict data from LLC
- Attacker can use huge pages
- Physical address selects the set to occupy
- Attacker evicts (fills set)



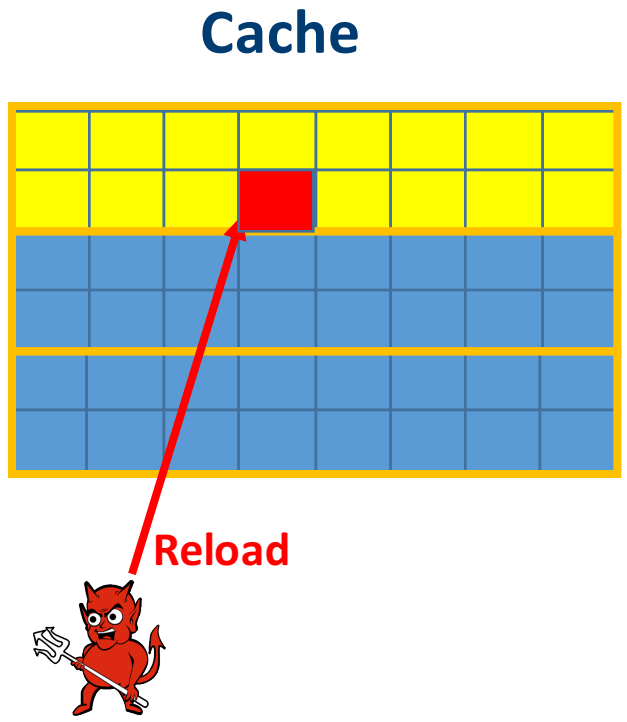
Evict + Reload Attack

- No flush instruction?
- Attacker needs to evict data from LLC
- Attacker can use huge pages
- Physical address selects the set to occupy
- Attacker evicts (fills set)
- Victim accesses/does not access



Evict + Reload Attack

- No flush instruction?
- Attacker needs to evict data from LLC
- Attacker can use huge pages
- Physical address selects the set to occupy
- Attacker evicts (fills set)
- Victim accesses/does not access
- Attacker reloads
  - Fast access time -> victim accessed
  - Slow access time -> victim did not access

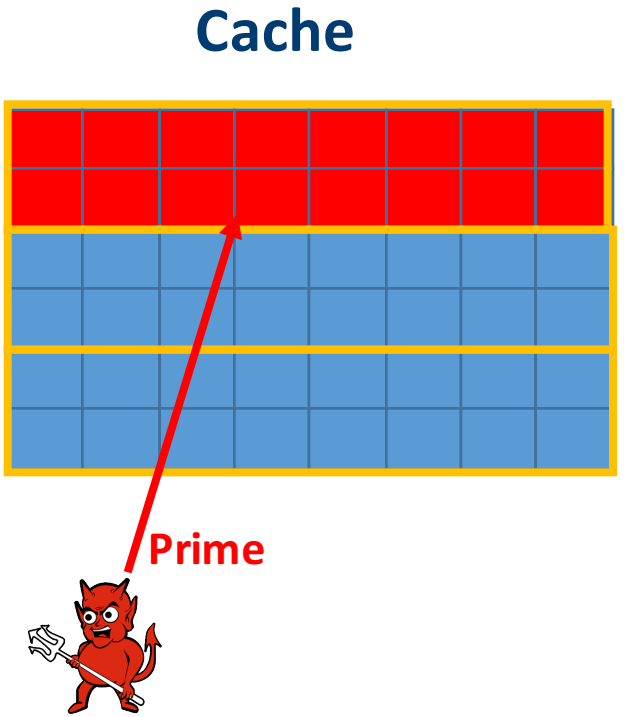


Evict + Reload Attack Summary

- Pros:
  - Applicable in processors without flush instruction (e.g. most ARM processors)
- Cons:
  - Can only target statically allocated memory
  - Deal with LLC slices (undocumented)
  - Only works with inclusive caches
  - Only works in the same CPU socket

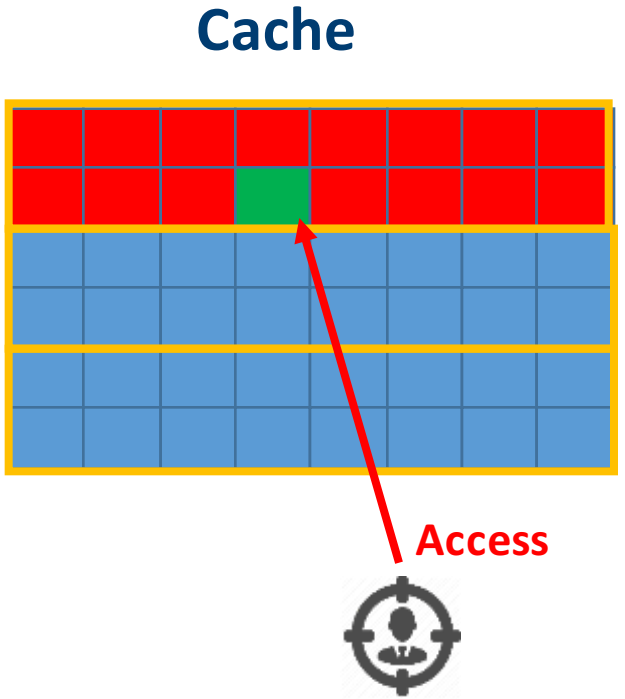
Prime + Probe Attack

- No shared memory pages?
  - Attacker can know the set utilized by the victim
- Attacker Primes



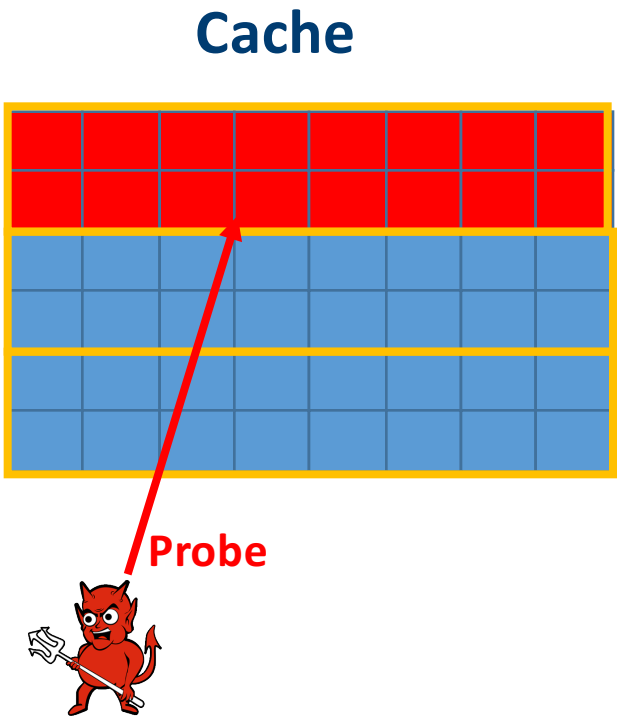
Prime + Probe Attack

- No shared memory pages?
  - Attacker can know the set utilized by the victim
- Attacker Primes
- Victim accesses/not accesses



Prime + Probe Attack

- No shared memory pages?
  - Attacker can know the set utilized by the victim
- Attacker Primes
- Victim accesses/not accesses
- Attacker re-accesses
  - Fast access time -> victim accessed
  - Slow access time -> victim did not access



Prime + Probe Attack Summary

- Pros
  - Does not need shared memory! (Broader impact)
  - Can target static and dynamically allocated memory!
- Cons:
  - Noisier than Flush + Reload
  - Dealing with LLC slices (undocumented)
  - Only works with inclusive caches
  - Only works in the same CPU socket
  - Need to identify the target set

How to retrieve information?

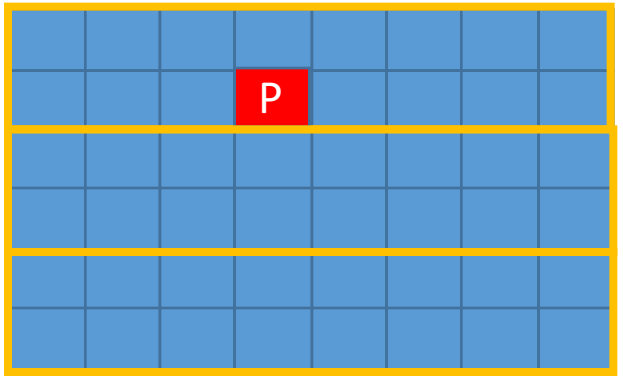
Montgomery ladder RSA

```
1  function modpow (a, b);
   Input  : base b, modulus N, secret
            E = (e_{k-1}, ..., e_1, e_0)
   Output: b^E mod N
2  R_0 = 1; R_1 = b;
3  for i = k - 1 downto 0 do
4      if e_i==0 then
5          R_1 = R_0 * R_1 mod N;
6          R_0 = R_0 * R_0 mod N;
7      end
8      else
9          R_0 = R_0 * R_1 mod N;
10         R_1 = R_1 * R_1 mod N;
11     end
12 end
13 return R_0;
```

Physical address

P=0x7fffc480

Flush and Reload Cache



How to retrieve information?

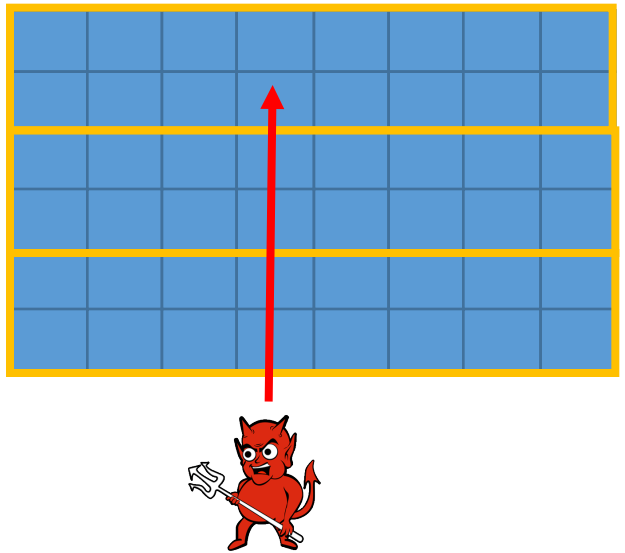
Montgomery ladder RSA

```
1  function modpow (a, b);
   Input  : base b, modulus N, secret
            E = (e_{k-1}, ..., e_1, e_0)
   Output: b^E mod N
2  R_0 = 1; R_1 = b;
3  for i = k - 1 downto 0 do
4      if e_i==0 then
5          R_1 = R_0 * R_1 mod N;
6          R_0 = R_0 * R_0 mod N;
7      end
8      else
9          R_0 = R_0 * R_1 mod N;
10         R_1 = R_1 * R_1 mod N;
11     end
12 end
13 return R_0;
```

Physical address

P=0x7fffc480

Flush and Reload Cache



How to retrieve information?

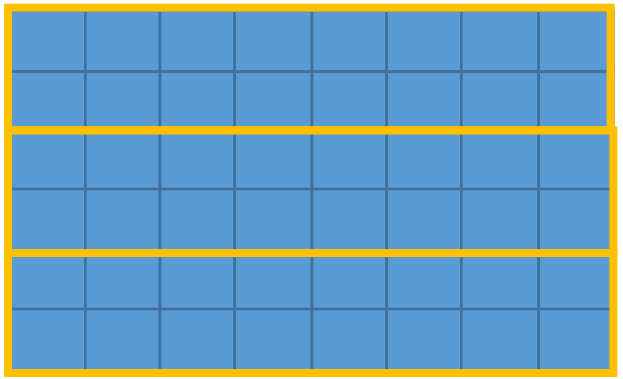
Montgomery ladder RSA

```
1  function modpow (a, b);
   Input  : base b, modulus N, secret
            E = (e_{k-1}, ..., e_1, e_0)
   Output: b^E mod N
2  R_0 = 1; R_1 = b;
3  for i = k - 1 downto 0 do
4      if e_i==0 then
5          R_1 = R_0 * R_1 mod N;
6          R_0 = R_0 * R_0 mod N;
7      end
8      else
9          R_0 = R_0 * R_1 mod N;
10         R_1 = R_1 * R_1 mod N;
11     end
12 end
13 return R_0;
```

Physical address

P=0x7fffc480

Flush and Reload Cache



How to retrieve information?

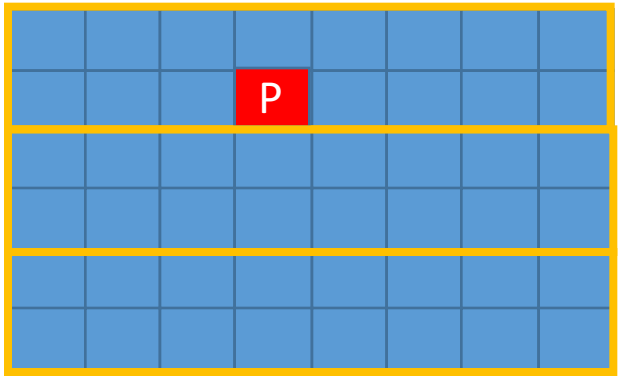
Montgomery ladder RSA

```
1  function modpow (a, b);
   Input  : base b, modulus N, secret
            E = (e_{k-1}, ..., e_1, e_0)
   Output: b^E mod N
2  R_0 = 1; R_1 = b;
3  for i = k - 1 downto 0 do
4      if e_i==0 then
5          R_1 = R_0 * R_1 mod N;
6          R_0 = R_0 * R_0 mod N;
7      end
8      else
9          R_0 = R_0 * R_1 mod N;
10         R_1 = R_1 * R_1 mod N;
11     end
12 end
13 return R_0;
```

Physical address

P=0x7fffc480

Flush and Reload Cache



How to retrieve information?

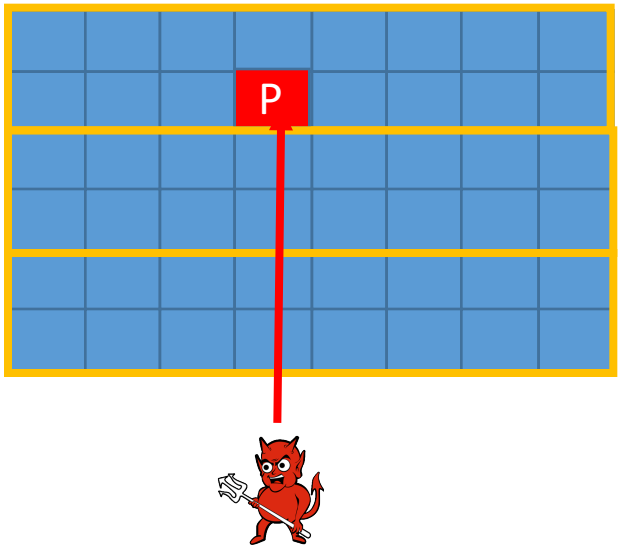
Montgomery ladder RSA

```
1  function modpow (a, b);
   Input  : base b, modulus N, secret
            E = (e_{k-1}, ..., e_1, e_0)
   Output: b^E mod N
2  R_0 = 1; R_1 = b;
3  for i = k - 1 downto 0 do
4      if e_i==0 then
5          R_1 = R_0 * R_1 mod N;
6          R_0 = R_0 * R_0 mod N;
7      end
8      else
9          R_0 = R_0 * R_1 mod N;
10         R_1 = R_1 * R_1 mod N;
11     end
12 end
13 return R_0;
```

Physical address

P=0x7fffc480

Flush and Reload Cache



How to retrieve information?

Montgomery ladder RSA

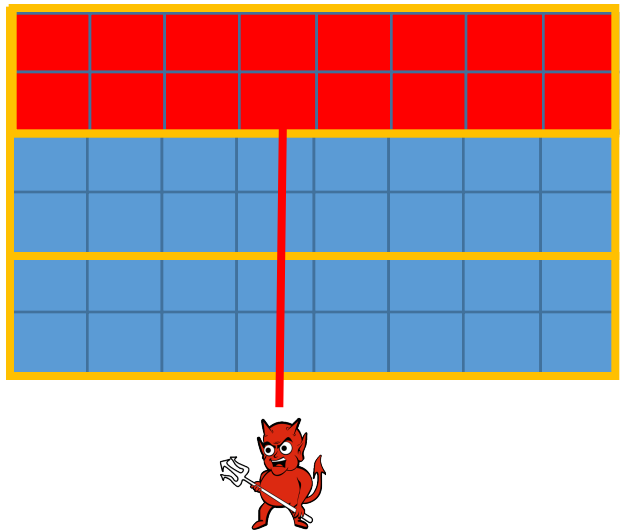
```
1  function modpow (a, b);
   Input  : base b, modulus N, secret
            E = (e_{k-1}, ..., e_1, e_0)
   Output: b^E mod N
2  R_0 = 1; R_1 = b;
3  for i = k - 1 downto 0 do
4      if e_i==0 then
5          R_1 = R_0 * R_1 mod N;
6          R_0 = R_0 * R_0 mod N;
7      end
8      else
9          R_0 = R_0 * R_1 mod N;
10         R_1 = R_1 * R_1 mod N;
11     end
12 end
13 return R_0;
```

Physical address

P=0x7fffc480

Prime and Probe

Cache



How to retrieve information?

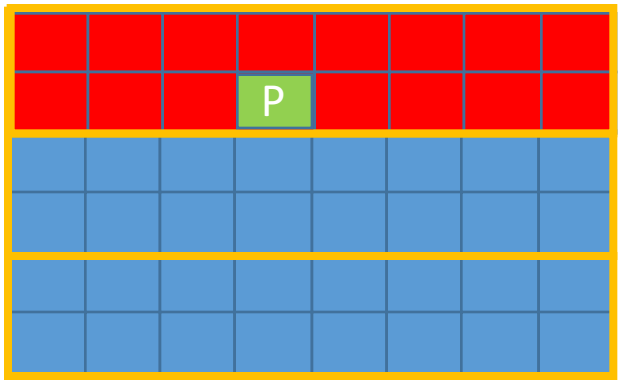
Montgomery ladder RSA

```
1  function modpow (a, b);
   Input  : base b, modulus N, secret
            E = (e_{k-1}, ..., e_1, e_0)
   Output: b^E mod N
2  R_0 = 1; R_1 = b;
3  for i = k - 1 downto 0 do
4      if e_i==0 then
5          R_1 = R_0 * R_1 mod N;
6          R_0 = R_0 * R_0 mod N;
7      end
8      else
9          R_0 = R_0 * R_1 mod N;
10         R_1 = R_1 * R_1 mod N;
11     end
12 end
13 return R_0;
```

Physical address

P=0x7fffc480

Prime and Probe Cache



How to retrieve information?

Montgomery ladder RSA

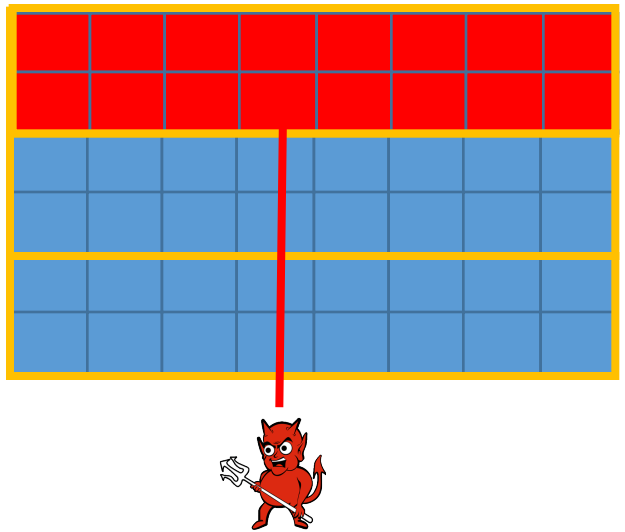
```
1  function modpow (a, b);
   Input  : base b, modulus N, secret
            E = (e_{k-1}, ..., e_1, e_0)
   Output: b^E mod N
2  R_0 = 1; R_1 = b;
3  for i = k - 1 downto 0 do
4      if e_i==0 then
5          R_1 = R_0 * R_1 mod N;
6          R_0 = R_0 * R_0 mod N;
7      end
8      else
9          R_0 = R_0 * R_1 mod N;
10         R_1 = R_1 * R_1 mod N;
11     end
12 end
13 return R_0;
```

Physical address

P=0x7fffc480

Prime and Probe

Cache



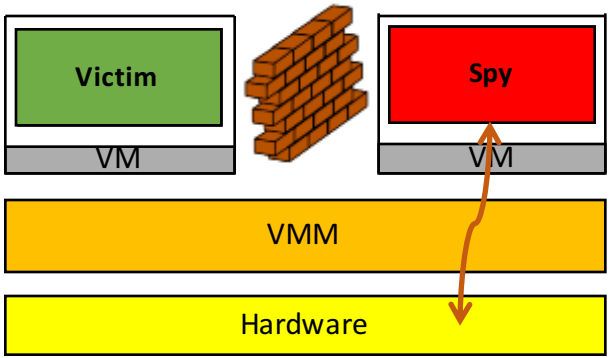
Attack Comparison

	Flush + Reload	Evict + Reload	Prime + Probe
Require Memory Deduplication	Y	Y	N
Require flush instruction	Y	N	N
Attack memory type	static	static	static + dynamic
Noise	low	low	high

Applicability

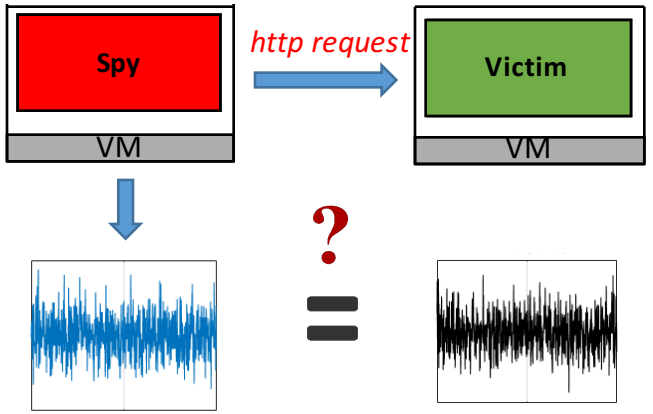
IaaS/PaaS Cloud Infrastructures

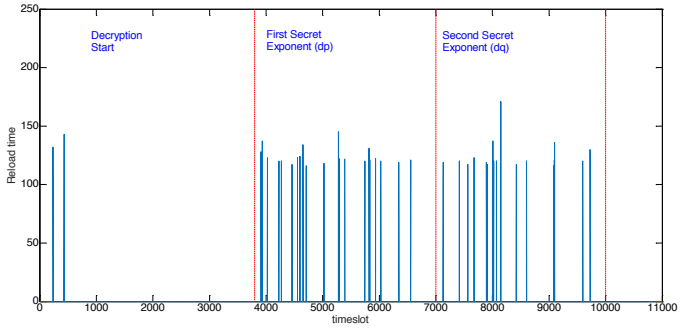
- VMs share underlying hardware
- Hardware isolation is usually not provided
- Example RSA in Amazon EC2 [INCI16]
- Pros:
  - Own virtualized OS. Access to timers or huge pages
  - If deduplication enabled, both attacks are applicable
- Cons:
  - Requires co-residency of VMs
  - High amount of noise



IaaS/PaaS Cloud Infrastructures

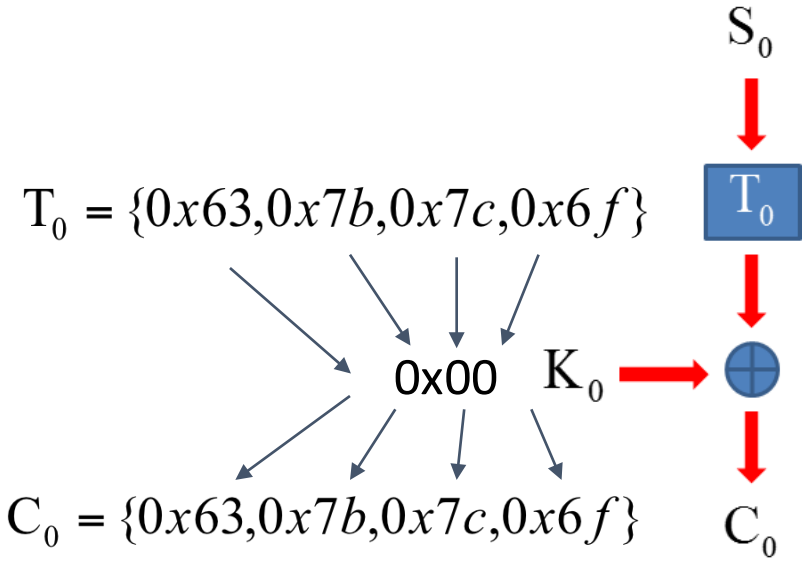
- How to find co-residency?
- Use available information!
- Profile the target LLC accesses
- Does the cache trace match the trace we expect?
  - If yes, co-residency
  - If no, open more VMs
- Other mechanisms utilize memory bus locking attacks
- Example RSA exponentiations easily distinguishable





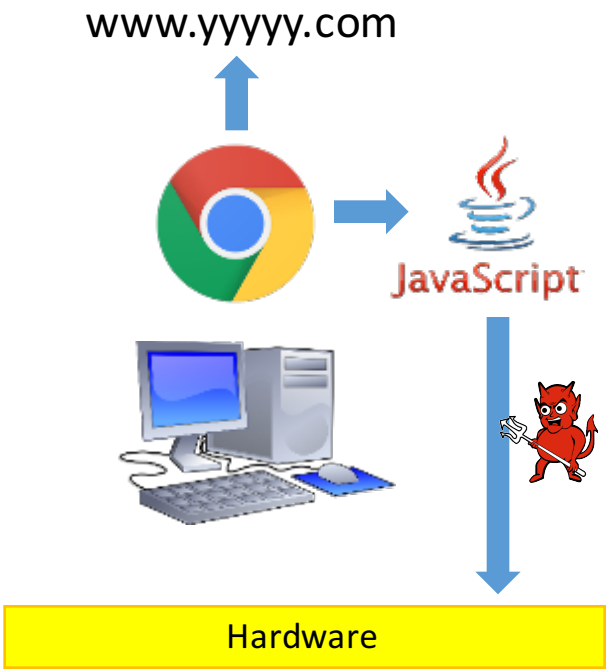
Demo: AES Key Recovery Across VMs

- We utilize KVM hypervisor
- Server using T-table AES (T-tables shared)
- Server encrypting plaintext with unknown key
- Attacker requests decryptions and recovers the key
- We check whether the entries of the T-tables have been used
- We XOR with the ciphertext after doing statistics to get the key



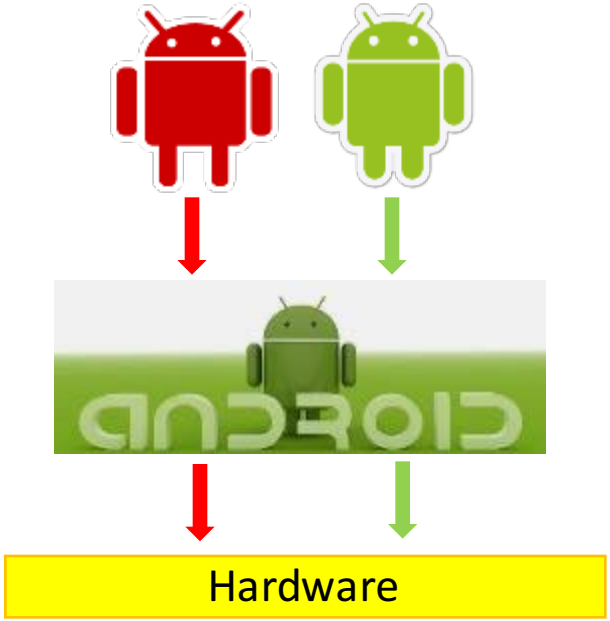
Browser Javascript

- Attacker embeds JS into the website
- Victim accesses the website
- Victim's browser executes the JS
- Example: Incognito browsing profiling [OREN15]
- Pros:
  - No need to find co-resident target
  - Attack executed in local machine (although sandboxed)
- Cons:
  - Flush and Reload can not be applied
  - Fine grain timers hard to achieve



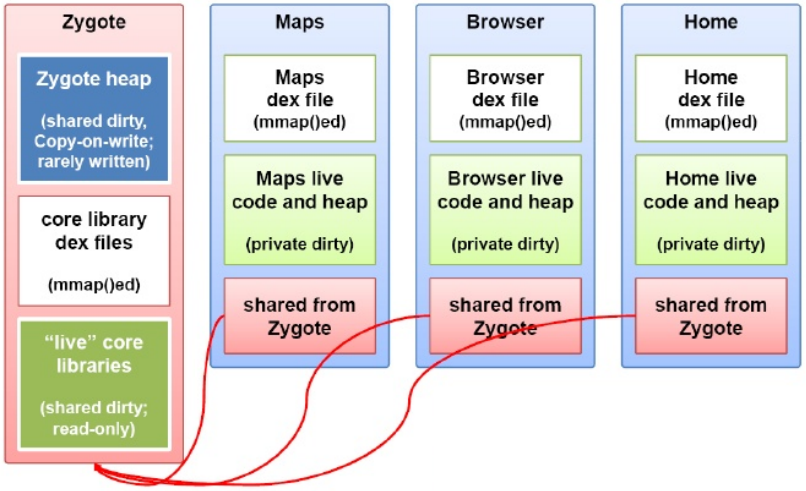
Smart Phone Applications

- Smartphone applications are logically isolated by the OS
- However, as with TEEs, all applications utilize the hardware caches
- Micro-architectural attacks look as innocent binaries, as they only perform timed memory accesses
- Example: AES key steal across apps [LIPP16]



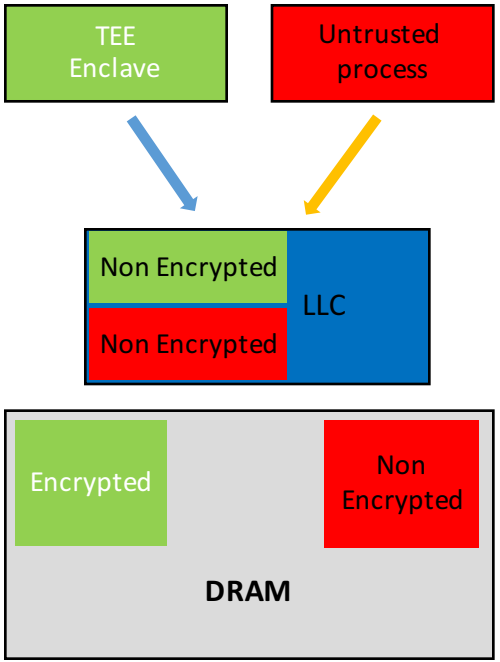
Smart Phone Applications

- Pros:
  - Deduplication is generally used (e.g. Android)
  - Easy deployment
- Cons
  - Flush instruction has to be enabled by SoC (only Samsun S6 for now)
  - Pseudo Random Replacement policies (reverse engineered)
  - Device dependent algorithms (e.g. non-inclusive caches or lockdown)



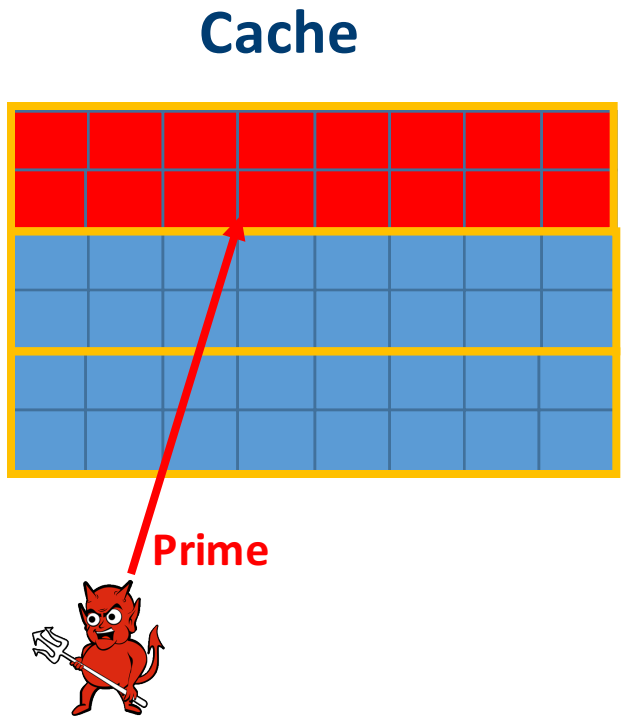
Trusted Execution Environment

- Trusted execution environments designed to achieve isolation from untrusted processes
- But both trusted and untrusted environments access same hardware caches!
- Enclave to enclave or host to enclave attacks are possible
- Example: TrustZone AES key steal [BRM15]
- Example: Intel SGX RSA key steal [SCW17]



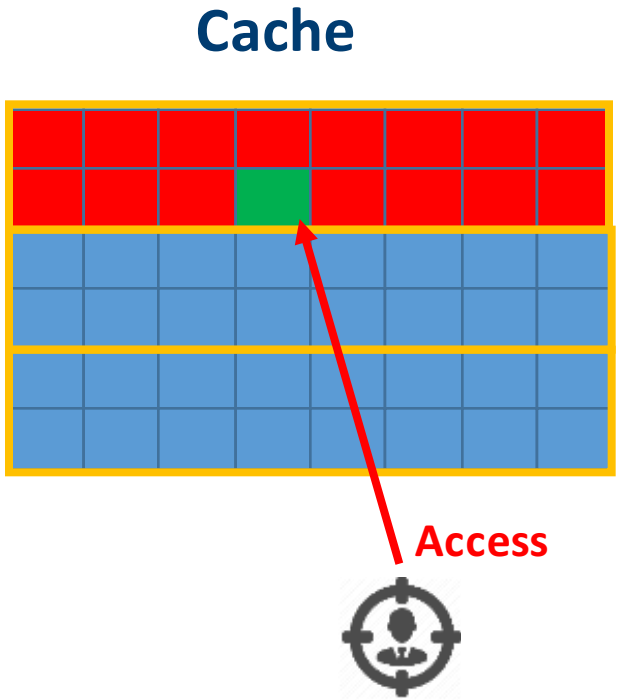
Trust Execution Environment

- Pros
  - Higher resolution: The OS can be malicious! more fine grain resources (including scheduling)
  - No need to find co-resident target
  - Limited noise: malicious OS can interrupt processes after (virtually) every memory access
- Cons
  - Flush and Reload not applicable (deduplication disabled)



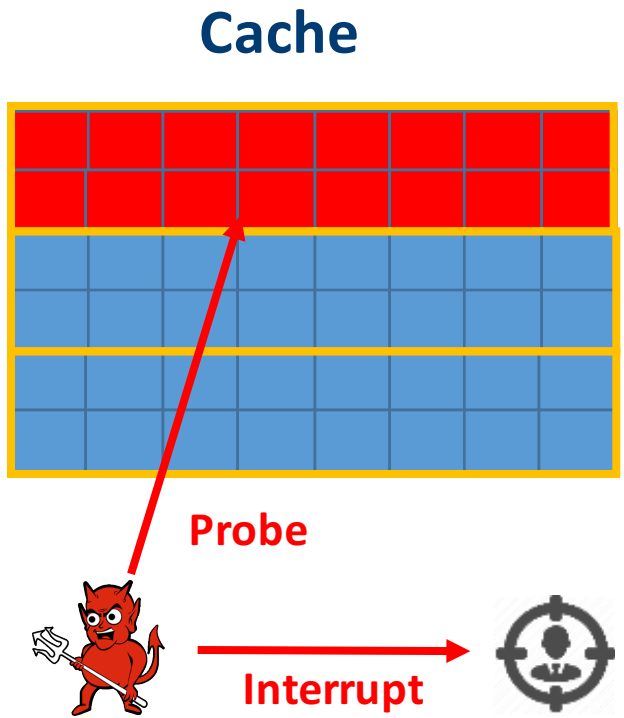
Trust Execution Environment

- Pros
  - Higher resolution: The OS can be malicious! more fine grain resources (including scheduling)
  - No need to find co-resident target
  - Limited noise: malicious OS can interrupt processes after (virtually) every memory access
- Cons
  - Flush and Reload not applicable (deduplication disabled)



Trust Execution Environment

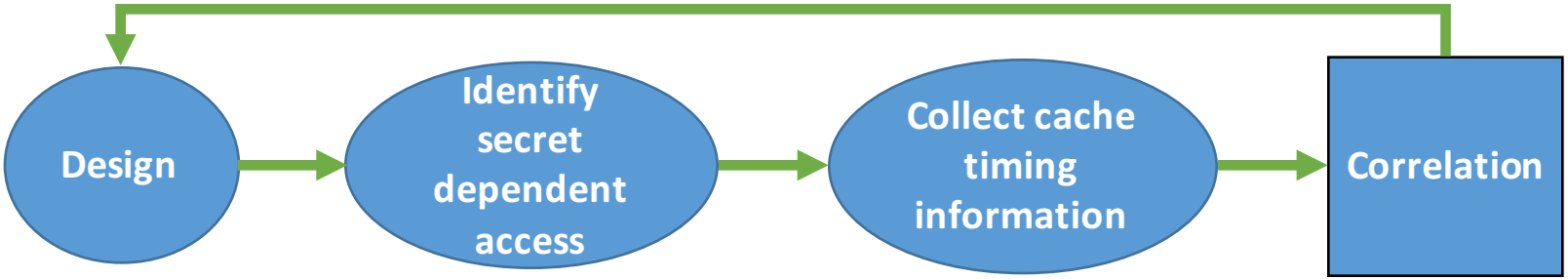
- Pros
  - Higher resolution: The OS can be malicious! more fine grain resources (including scheduling)
  - No need to find co-resident target
  - Low noise: malicious OS can interrupt processes after (virtually) every memory access
- Cons
  - Flush and Reload not applicable (deduplication disabled)



Countermeasures

Design Cache Leakage Free Code

- Secret independent instruction accesses
- Secret independent data accesses
- Identification of variables that contain information related to the secret (manual inspection, taint analysis, etc.)
- Obtain cache timing traces to correlate with the secret variables to measure the leakage



Design Cache Leakage Free Code

```
1  function modpow (a, b);
   Input  : base b, modulus N, secret
            E = (e_{k-1}, ..., e_1, e_0)
   Output: b^E mod N
2  R_0 = 1; R_1 = b;
3  for i = k − 1 downto 0 do
4      if e_i==0 then
5          R_1 = R_0 * R_1 mod N;
6          R_0 = R_0 * R_0 mod N;
7      end
8      else
9          R_0 = R_0 * R_1 mod N;
10         R_1 = R_1 * R_1 mod N;
11     end
12 end
13 return R_0;
```

CVE-2016-7439

Design Cache Leakage Free Code

```
1  function modpow (a, b);
   Input  : base b, modulus N, secret
            E = (e_{k-1}, ..., e_1, e_0)
   Output: b^E mod N
2  R_0 = 1; R_1 = b;
3  for i = k − 1 downto 0 do
4      if e_i==0 then
5          R_1 = R_0 * R_1 mod N;
6          R_0 = R_0 * R_0 mod N;
7      end
8      else
9          R_0 = R_0 * R_1 mod N;
10         R_1 = R_1 * R_1 mod N;
11     end
12 end
13 return R_0;
```

CVE-2016-7439

secret dependent instruction access

Design Cache Leakage Free Code

CVE-2016-7439

```
1 function modpow (a, b);
  Input  : base b, modulus N, secret
           E = (e_{k-1}, ..., e_1, e_0)
  Output: b^E mod N
2 R[0] = 1; R[1] = b;
3 for i = k - 1 downto 0 do
4 |   R[ê_i] = R[0] * R[1] mod N;
5 |   R[e_i] = R[e_i] * R[e_i] mod N;
6 end
7 return R[0];
```

Secret **independent** instruction access

Design Cache Leakage Free Code

CVE-2016-7439

```
1 function modpow (a, b);
  Input  : base b, modulus N, secret
           E = (e_{k-1}, ..., e_1, e_0)
  Output: b^E mod N
2 R[0] = 1; R[1] = b;
3 for i = k − 1 downto 0 do
4 |   R[ê_i] = R[0] * R[1] mod N;
5 |   R[e_i] = R[e_i] * R[e_i] mod N;
6 end
7 return R[0];
```

Secret **independent** instruction access

Secret dependent data access

Design Cache Leakage Free Code

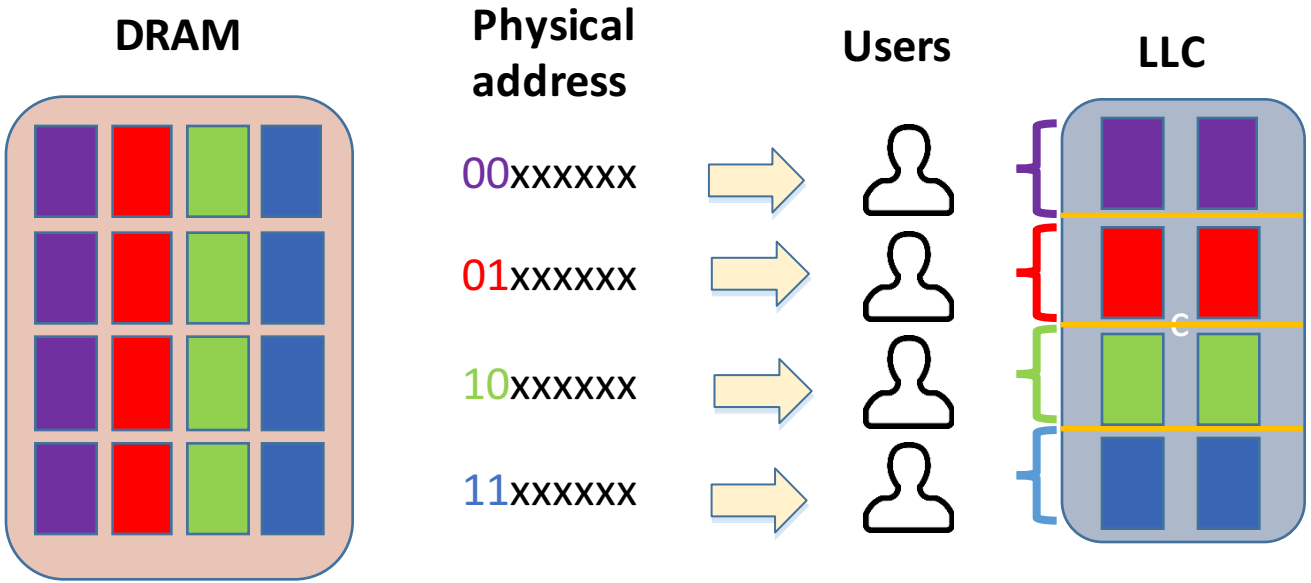
```
1 function modpow (a, b);
  Input  : base b, modulus N, secret
           E = (e_{k-1}, ..., e_1, e_0)
  Output: b^E mod N
2 R[0] = 1; R[1] = b;
3 for i = k − 1 downto 0 do
4 |   R[0] * e_i + R[1] * ê_i = R[0] * R[1] mod N;
5 |   R[1] * e_i + R[0] * ê_i =
  |     R[1] * R[1] * e_i + R[0] * R[0] * ê_i mod N;
6 end
7 return R[0];
```

Secret **independent** instruction access

Secret **independent** data access

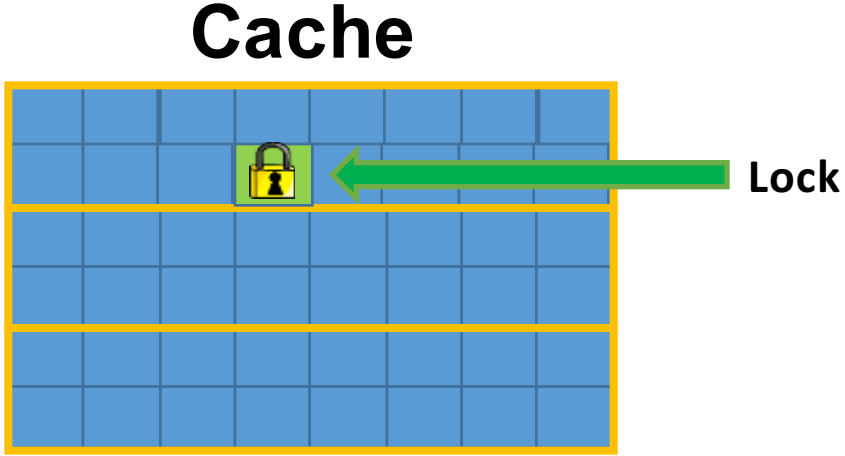
Page Coloring

- Avoiding collisions in the LLC
- Location in LLC determined by physical address
- Give each user a color (address bits)



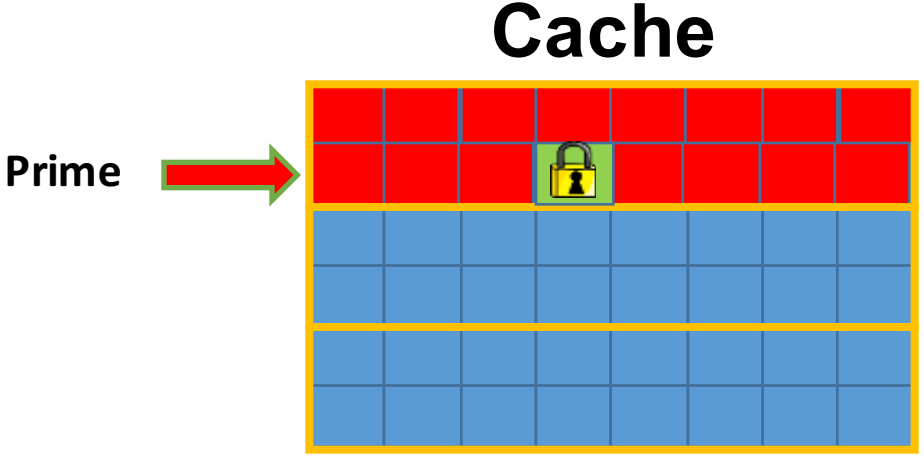
Cache Allocation Technology

- Intel CAT provides hardware framework to lock the cache
- Allows OS/hypervisor to mark cache ways as un-evictable
- Attacker can not influence victim's cache accesses
- Modify hypervisor to support more lock partitions [LIU16]



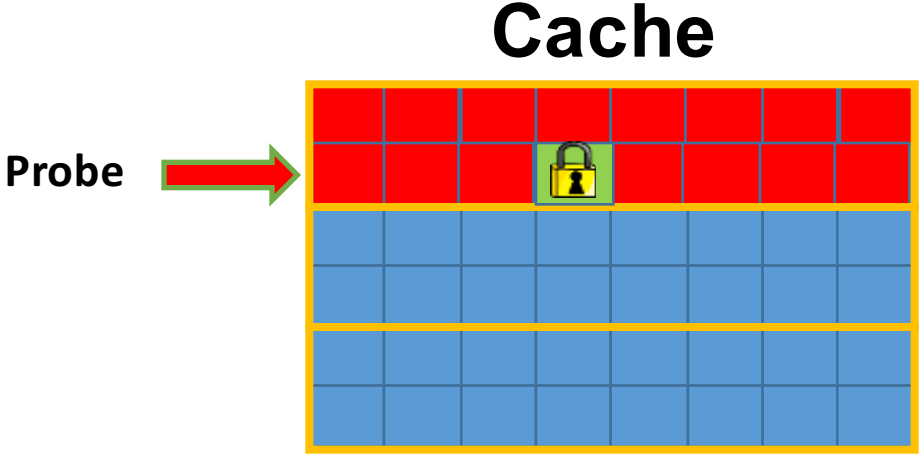
Cache Allocation Technology

- Intel CAT provides hardware framework to lock the cache
- Allows OS/hypervisor to mark cache ways as un-evictable
- Attacker can not influence victim's cache accesses
- Modify hypervisor to support more lock partitions [LIU16]



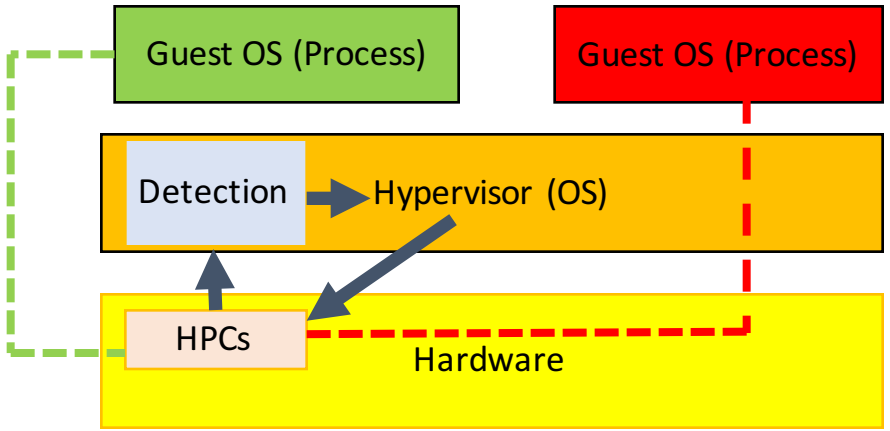
Cache Allocation Technology

- Intel CAT provides hardware framework to lock the cache
- Allows OS/hypervisor to mark cache ways as un-evictable
- Attacker can not influence victim's cache accesses
- Modify hypervisor to support more lock partitions [LIU16]



Behavior Detection

- Hardware Performance Counters (HPCs) can track hardware events (e.g. LLC misses)
- LLC attacks leave a clear trace in terms of cache misses/hits
- Hypervisor/OS tracks this events to detect unusual behavior
- Detection can be improved by inspecting memory access



Countermeasure Comparison (Requirements)

	Leakage Free Code	Page Coloring	Intel CAT	Behavior Detection
Require source code change	Y	N	N	N
Require OS (hypervisor) update	N	Y	Y	Depends
Require new hardware	N	N	Y	N

Countermeasure Comparison (Coverage)

	Leakage Free Code	Page Coloring	Intel CAT	Behavior Detection
IaaS/PaaS	Y	Y	Depends	Y
Javascript in broswer	Y	Depends	Depends	Y
Smartphone	Y	Y	Depends	Y
TEE	Y	N	N	N

Key Takeaways

- Cache attacks are complex but a real threat!
- Flush+Reload, Evict+Reload, Prime+Probe
- IaaS/PaaS, web browsers, smartphones, TEE,...What else?
- Call to action:
  - Application level: introduce cache leakage free code design
  - Hypervisor/OS level: page coloring for cache isolation
  - System level: use software to leverage hardware features (Intel CAT, performance counters)

References

[INCI16] Inci,M., Gulmezoglu, B., Irazoqui, G., Eisenbarth, T., Sunar, B. *Cache Attacks Enable Bulk Key Recovery on the Cloud.* CHES 2016

[OREN15] Oren,Y., Kemerlis, V., Sethumadhavan, S, Keromytis, A. *The Spy in the Sandbox: Practical Cache Attacks in JavaScript and their Implications.* ACM CCS 2015

[BRM15] Brumley,B. *Cache Storage Attacks.* CT-RSA 2015

[SCW17] Schwarz,M., Weiser,S., Gruss, D., Maurice, C., Mangard, S. *Malware Guard Extension: Using SGX to Conceal Cache Attacks.* Arxiv 2017

[LIPP16] Lipp,M., Gruss, D., Spreitzer, R., Maurice, C., Mangard, S. *ARMageddon: Cache Attacks on Mobile Devices.* USENIX 2016

[LIU16] Liu, F., Yarom, Y., Mckeen, F., Rozas, C., Heiser, G., Lee R. CATalyst: Defeating last-level cache side channel attacks in cloud computing. HPCA 2016