DAWG: A Defense Against Cache Timing Attacks in Speculative Execution Processors

Vladimir Kiriansky, Ilia Lebedev, Saman Amarasinghe, Srinivas Devadas, Joel Emer
{vlk, ilebedev, saman, devadas, emer}@csail.mit.edu

MICRO'18
October 24, 2018
Fukuoka, Japan
Outline

• Cache access timing attacks
• DAWG protection mechanism: Cache, Core
• OS support: System Calls, Resource Management
• Performance and security evaluation
• Conclusion & Q/A
Trust Boundaries
Trust Boundaries

- Hypervisor
- OS
- Process
- Sandbox
- Enclave
Trust Boundary Crossing APIs / Attack Vectors

Legal API

Hypervisor

OS

Process

Sandbox

Process

Enclave
Trust Boundary Crossing APIs / Attack Vectors

Legal API

Illegal Channel

OS

Hypervisor

Enclave

Process

Sandbox

APIs

Attack Vectors
Side Channels and Covert Channels

Victim's Protection Domain

Secret data

Attacker's Protection Domain

Stolen data
Side Channels and Covert Channels

- **Accessor**
  - Existing code - non-speculative, traditional
  - Synthesized - Spectre 1.0, 1.1 - unresolved
Side Channels and Covert Channels

• Accessor
  - Existing code - non-speculative, traditional
  - Synthesized - Spectre 1.0, 1.1 - unresolved
Side Channels and Covert Channels

- **Accessor**
  - Existing code - non-speculative, traditional
  - Synthesized - Spectre 1.0, 1.1 - unresolved

- **Channel** = micro-architectural state: cache, TLB, branch predictor state, etc.

Victim's Protection Domain
- Secret data
- Accessor
- Transmitter

Attacker's Protection Domain
- Receiver
- Stolen data

covert channel

Stolen data

Secret data

Victim's Protection Domain

Accessor

Transmitter

Stolen data

Receiver
Side Channels and Covert Channels

- **Accessor**
  - Existing code - non-speculative, traditional
  - Synthesized - Spectre 1.0, 1.1 - unresolved

- **Channel** = micro-architectural state: `cache`, TLB, branch predictor state, etc.
Cache Covert Channel

Victim's Protection Domain
Transmitter

Attacker's Protection Domain
Receiver

cache covert channel
Cache Covert Channel: Shared Cache Ways
Cache Covert Channel: Shared Cache Ways

[Flush+Reload, Evict+Reload, Thrash+Reload]

1. Receiver evicts block A
Flush / Evict / Thrash
Cache Covert Channel: Shared Cache Ways

1. Receiver evicts block A
Flush / Evict / Thrash

2-way Cache Set
Cache Covert Channel: Shared Cache Ways

1. Receiver evicts block A
   Flush / Evict / Thrash

2. Transmitter sends a 0 or 1 secret bit via access to A
Cache Covert Channel: Shared Cache Ways

1. Receiver evicts block A
   Flush / Evict / Thrash

2. Transmitter sends a 0 or 1 secret bit via access to A

3. Receiver times access to A
Cache Covert Channel: Shared Cache Ways

1. Receiver evicts block A
   Flush / Evict / Thrash

2. Transmitter sends a 0 or 1 secret bit via access to A

3. Receiver times access to A
   infers secret bit ♠️
Cache Covert Channel

Victim

Transmitter

Attacker

Receiver

cache covert channel
Block Cache Covert Channel?

Victim

Attacker

Transmitter

Receiver
DAWG: Dynamically Allocated Way Guard

- Cache Protection Domains
- Non-interference by any action: hit / flush / eviction / fill
DAWG: Dynamically Allocated Way Guard

- Cache Protection Domains
- Non-interference by any action: hit / flush / eviction / fill
- Partitioned ways of set-associative structures
  - Domain-private cache tag state
DAWG: Dynamically Allocated Way Guard

- Cache Protection Domains
- Non-interference by any action: hit / flush / eviction / fill
- Partitioned ways of set-associative structures
  - Domain-private cache tag state
  - Domain-private replacement metadata
No Cache Covert Channel: Private Cache Ways
No Cache Covert Channel: Private Cache Ways

1. Receiver evicts block A?
   Flush / Evict / Thrash
No Cache Covert Channel: Private Cache Ways

1. Receiver evicts block A
   Flush / Evict / Thrash
No Cache Covert Channel: Private Cache Ways

1. Receiver evicts block A
   Flush / Evict / Thrash

2. Transmitter sends a 0 or 1 secret bit via access to A
No Cache Covert Channel: Private Cache Ways

1. Receiver evicts block A
   - Flush / Evict / Thrash

2. Transmitter sends a 0 or 1 secret bit via access to A

3. Receiver times access to A
No Cache Covert Channel: Private Cache Ways

1. Receiver evicts block A
   Flush / Evict / Thrash

2. Transmitter sends a 0 or 1 secret bit via access to A

3. Receiver times access to A
   no signal
No Cache Covert Channel: Private Cache Ways

1. Receiver evicts block A
   Flush / Evict / Thrash

2. Transmitter sends a 0 or 1 secret bit via access to A

3. Receiver times access to A

   no signal

   ⚫ 0  ➔  ⚫ 1
No Cache Covert Channel

Victim Domain

Attacker Domain

Transmitter

Receiver

そうでない場合

DAWG
CAT: QoS Cache Partitioning

• Starting point in production silicon: Intel's Cache Allocation Technology for LLC

• Iyer et al [SC'04, SIGMETRICS'07, MICRO'07] From concept to reality in Haswell [HPCA'16]

• Not a security barrier

Quality of Service goal: prevent one application from dominating the cache
CAT: Way-Partitioned Set-associative Caches

- Way-partitioning LLC
- Protection domain IDs
- Fill mask
DAWG: Dynamically Allocated Way Guard

- Way-partitioning L1-L3
- Protection domain IDs
- Fill mask
DAWG: Dynamically Allocated Way Guard

- Way-partitioning **L1-L3**
- Protection domain IDs
  - Fill mask
  - Hit mask
    - Hits
DAWG: Dynamically Allocated Way Guard

- Way-partitioning L1-L3
- Protection domain IDs
  - Fill mask
  - Hit mask
    - Hits
    - PLRU updates

Cache controller state machine

- Address
- Tag
- Set Index
- Way write enables
- Cache line write data
- Coherence logic
- Fill isolation
- Metadata isolation
- Hit isolation
- Policies
Higher Security than QoS Cache Partitioning

- Production QoS way-partitioning (CAT) by design allows hits across domains
- Not a security barrier

<table>
<thead>
<tr>
<th></th>
<th>CAT</th>
<th>DAWG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Way allocation</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hits in victim</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
Higher Security than QoS Cache Partitioning

- Production QoS way-partitioning (CAT) by design allows hits across domains
- Not a security barrier

<table>
<thead>
<tr>
<th></th>
<th>CAT</th>
<th>DAWG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Way allocation</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Hits in victim</td>
<td>❌</td>
<td>✅</td>
</tr>
<tr>
<td>Flush in victim</td>
<td>❌</td>
<td>✅</td>
</tr>
</tbody>
</table>

Flush
Higher Security than QoS Cache Partitioning

- Production QoS way-partitioning (CAT) by design allows hits across domains
- Not a security barrier

<table>
<thead>
<tr>
<th></th>
<th>CAT</th>
<th>DAWG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Way allocation</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Hits in victim</td>
<td>❌</td>
<td>✔️</td>
</tr>
<tr>
<td>Flush in victim</td>
<td>❌</td>
<td>✔️</td>
</tr>
<tr>
<td>PLRU/NRU leak</td>
<td>❌</td>
<td>✔️</td>
</tr>
</tbody>
</table>
## Shared Memory ↔ Shared Cache

<table>
<thead>
<tr>
<th></th>
<th>CAT</th>
<th>DAWG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hits in victim</td>
<td>❌</td>
<td>✅</td>
</tr>
<tr>
<td>Flush in victim</td>
<td>❌</td>
<td>✅</td>
</tr>
</tbody>
</table>

Flush+Reload
Evict+Reload
Thrash+Reload
# Shared Sets ↔ Shared Metadata

<table>
<thead>
<tr>
<th></th>
<th>CAT</th>
<th>DAWG</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLRU/NRU leak</td>
<td>❌</td>
<td>✅</td>
</tr>
</tbody>
</table>

PLRU-Prime+Probe
OS Support and Resource Management
Protection Domain Isolation

Legal API

Illegal Channels

OS

Hypervisor

Enclave

Sand box

Process

Process

OS
Protection Domain Isolation

Legal API

Illegal Channels

Hypervisor

OS

Enclave

Process

Sandbox

Process
Protection Domain Isolation

Legal API

Hypervisor

OS

Process

Process

Sand box

Enclave
Fast System Calls

1. OS can access everything in process memory
2. In/out arguments in cache (dirty)
3. OS must not leak
Core & OS changes: Domain Descriptors

- Existing support for CAT
Core & OS changes: Domain Descriptors

- Existing support for CAT + DAWG

Fill Mask
Hit Mask

Domain Descriptors
Global

0 0111, 0111
1 1000, 1000
Core & OS changes: Domain Selectors

- Existing support for SMAP (Supervisor Mode Access Protection)

  Few routines access user-data & toggle SMAP
  
  ```c
  copy_from_user
  copy_to_user
  ...
  ```
Core & OS changes:
Domain Selectors

• Existing support for SMAP + DAWG

• Core MSR: separate code / load / store selectors
Core & OS changes: System calls

- Existing support for CAT & SMAP + DAWG

- Core MSR: separate code / load / store selectors

Fill Mask

Hit Mask

Code: User
Load: User
Store: User

Domain Selectors Per-Thread
Core & OS changes: System calls

• Existing support for CAT & SMAP + DAWG

• Core MSR: separate code / load / store selectors

```
copy_from_user
```
Core & OS changes: System calls

- Existing support for CAT & SMAP + DAWG
- Core MSR: separate code / load / store selectors

```
copy_to_user
```
Resource Management

- Extends CAT support + secure domain reallocation
- Secure dynamic way reassignment

Fill Mask

Hit Mask
Secure Dynamic Way Reassignment

- Secure way sanitization
- Concurrent for shared caches

Fill Mask

Hit Mask

Flush blocks in revoked way
Secure Dynamic Way Reassignment

Fill Mask

Hit Mask
Secure Dynamic Way Reassignment

Fill Mask

Hit Mask
Secure Dynamic Way Reassignment

Fill Mask

Hit Mask
DAWG Beyond Cache Partitioning

- Cache Way Locking

Fill Mask
Hit Mask
Core & OS changes

• Shared libraries, memory mapped I/O, VM page sharing, and cache coherence

• Details in our paper
Matching Performance of QoS Cache Partitioning

- Typical use case: public cloud VM isolation (no page sharing, no core sharing, no SMT)

→ DAWG's performance is identical to production LLC way-partitioning (Intel's CAT)

 VM1       VM2
Way-Partitioning

- Cycles / Edge (K)
- 8/16 ways (fair share)
- 15/16 ways (1 way for OS)
- 16/16 ways (insecure baseline)

Graph Size (log N)

bc
pr
tc
bfs
cc
sssp

Power-law graphs [GAPBS]

[in zsim]
Shared Data: DAWG vs CAT

Shared read-only mapping

Graph Size (log N)

Slowdown

bc
pr
tc
bfs
cc
sssp

[on Haswell]
Security Evaluation
Cache Partitioning $\approx$ Dedicated Host Per Domain

Illegal Channels

Isolating peers
Cache Partitioning ≈ Dedicated Host Per Domain

Illegal Channels

Isolating peers
Cache Partitioning $\approx$ Dedicated Host Per Domain

Illegal Channels

Isolating peers and parents
Cache Partitioning ≈ Dedicated Host Per Domain

Secure API

Illegal Channels

Secure communication
Dedicated Host Insufficient: Remote Cache Timing Attacks

- High-bandwidth remote cache timing attack
Remote Cache Reflection: Attacks and Defenses

- High-bandwidth remote cache timing attack
Conclusion

- Partitioning is the foundation
- Minimal changes to hardware: Build on CAT
- Minimal changes to OS: Build on SMAP
- Minimal performance overhead: Zero or small over CAT QoS
- DAWG applies beyond caches: TLB, etc
Thanks

Vladimir Kiriansky
vlk@csail.mit.edu
Backup Slides
Beyond Cache Partitioning: Code Prioritization
Beyond Cache Partitioning Streaming Data Isolation

- Graph application use case:
  1-way for streaming edges
  3-ways for per-vertex data