SPYING ON YOUR NEIGHBOR: CPU CACHE ATTACKS AND BEYOND
We would like to protect against cache attacks generically

You won’t believe this one thing that people forget
OVERVIEW

- Side channels
- Cache attacks
- TLBleed
- Evaluation
CACHE SIDE CHANNELS
SIDE CHANNELS
SIDE CHANNELS

• Leak secrets outside the regular interface
SIDE CHANNELS

• Leak secrets outside the regular interface
SIDE CHANNELS

• Leak secrets outside the regular interface

• The first combination safes in the 1950s
EXAMPLE: FLUSH+RELOAD

• Can attack AES implementation with T tables

• A table lookup happens $T_j [x_i = p_i \oplus k_i ]$

• $p_i$ is a plaintext byte, $k_i$ a key byte

• We can detect lookups into the table using F+R
EXAMPLE: FLUSH+RELOAD

• Again: secrets are betrayed by memory accesses
• Known plaintext + accesses = key recovery
EXAMPLE: FLUSH+RELOAD

- Again: secrets are betrayed by memory accesses
- Known plaintext + accesses = key recovery
EXAMPLE: LIBGCRYPT ECC

- Not side channel proof version:
EXAMPLE: LIBGCRYPT ECC

- Not side channel proof version:

```c
void _gcry_mpi_ec_mul_point (mpi_point_t result,
   gcry_mpi_t scalar, mpi_point_t point,
   mpi_ec_t ctx)
{
    ...
    for (j=nbits-1; j >= 0; j--)
    { _gcry_mpi_ec_dup_point (result, result, ctx);
        if (mpi_test_bit (scalar, j))
        _gcry_mpi_ec_add_points(result,result,point,ctx);
    }
    ...
}
```
EXAMPLE: LIBGCRYPT ECC

• Not side channel proof version:
EXAMPLE: LIBGCRYPT ECC
EXAMPLE: LIBGCrypt ECC
EXAMPLE: LIBGCRYPT ETC

• More side channel proof version
EXAMPLE: LIBGCRYPT ETC
DEFENCE EXAMPLE: TSX
DEFENCE EXAMPLE: TSX

- Intel TSX: Transactional Synchronization Extensions
DEFENCE EXAMPLE: TSX

• Intel TSX: Transactional Synchronization Extensions
• Intended for hardware transactional memory
DEFENCE EXAMPLE: TSX

• Intel TSX: Transactional Synchronization Extensions
• Intended for hardware transactional memory
• But relies on unshared cache activity
DEFENCE EXAMPLE: TSX

- Intel TSX: Transactional Synchronization Extensions
- Intended for hardware transactional memory
- But relies on unshared cache activity
- Transactions fit in cache, otherwise auto-abort
DEFENCE EXAMPLE: TSX

• Intel TSX: Transactional Synchronization Extensions

• Intended for hardware transactional memory

• But relies on unshared cache activity

• Transactions fit in cache, otherwise auto-abort

• We can use this as a defence - all solved now right?
DEFENCE EXAMPLE: TSX

- Intel TSX: Transactional Synchronization Extensions
- Intended for hardware transactional memory
- But relies on unshared cache activity
- Transactions fit in cache, otherwise auto-abort
- We can use this as a defence - all solved now right?
DEFENCE EXAMPLE: TSX

- Intel TSX: Transactional Synchronization Extensions
- Intended for hardware transactional memory
- But relies on unshared cache activity
- Transactions fit in cache, otherwise auto-abort
- We can use this as a defence - all solved now right?
TLBLEED: TLB AS SHARED STATE
TLBLEED: TLB AS SHARED STATE

• Other structures than cache shared between threads?
• What about the TLB?
• Documented: TLB has L1iTLB, L1dTLB, and L2TLB
• Not documented: structure
TLB BLEED: TLB AS SHARED STATE
TLBLEED: TLB AS SHARED STATE

• Let’s experiment with performance counters

• Try linear structure first

• All combinations of ways (set size) and sets (stride)

• Smallest number of ways is it

• Smallest corresponding stride is number of sets
TLBLEED: TLB AS SHARED STATE

• Let’s experiment with performance counters

• Try linear structure first

• All combinations of ways (set size) and sets (stride)

• Smallest number of ways is it

• Smallest corresponding stride is number of sets
TLBLEED: TLB AS SHARED STATE

- Let’s experiment with performance counters
- Try linear structure first
- All combinations of ways (set size) and sets (stride)
- Smallest number of ways is it
- Smallest corresponding stride is number of sets
TLB BLEED: TLB AS SHARED STATE
TLBLEED: TLB AS SHARED STATE

• For L2TLB:
  We reverse engineered a more complex hash function
TL BLEED: TLB AS SHARED STATE

• For L2TLB:
  We reverse engineered a more complex hash function

• Skylake XORs 14 bits, Broadwell XORs 16 bits
TLBLEED: TLB AS SHARED STATE

- For L2TLB:
  We reverse engineered a more complex hash function

- Skylake XORs 14 bits, Broadwell XORs 16 bits

- Represented by this matrix, using modulo 2 arithmetic
TLBLEED: TLB AS SHARED STATE

• For L2TLB:
  We reverse engineered a more complex hash function

• Skylake XORs 14 bits, Broadwell XORs 16 bits

• Represented by this matrix, using modulo 2 arithmetic

\[
H = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]
TLBLEED: TLB AS SHARED STATE
TL BLEED: TLB AS SHARED STATE

- Let’s experiment with performance counters
TLBLEED: TLB AS SHARED STATE

• Let’s experiment with performance counters

• Now we know the structure..
  Are TLB’s shared between hyperthreads?
TLBLEED: TLB AS SHARED STATE

- Let’s experiment with performance counters
- Now we know the structure..
  Are TLB’s shared between hyperthreads?
- Let’s experiment with misses when accessing the same set
TLBLEED: TLB AS SHARED STATE

• Let’s experiment with performance counters

• Now we know the structure..
Are TLB’s shared between hyperthreads?

• Let’s experiment with misses when accessing the same set
TLBLEED: TLB AS SHARED STATE

• Let’s experiment with performance counters

• Now we know the structure..
  Are TLB’s shared between hyperthreads?

• Let’s experiment with misses when accessing the same set
TLBLEED: TLB AS SHARED STATE

• Let’s experiment with performance counters

• Now we know the structure..
  Are TLB’s shared between hyperthreads?

• Let’s experiment with misses when accessing the same set
TLBLEED: TLB AS SHARED STATE

<table>
<thead>
<tr>
<th>Name</th>
<th>year</th>
<th>L1 dTLB</th>
<th>L1 iTLB</th>
<th>L2 sTLB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sandybridge</td>
<td>2011</td>
<td>16 4 7.0 lin ✓</td>
<td>16 4 50.0 lin ✓</td>
<td>128 4 16.3 lin ✓</td>
</tr>
<tr>
<td>Ivybridge</td>
<td>2012</td>
<td>16 4 7.1 lin ✓</td>
<td>16 4 49.4 lin ✓</td>
<td>128 4 18.0 lin ✓</td>
</tr>
<tr>
<td>Haswell</td>
<td>2013</td>
<td>16 4 8.0 lin ✓</td>
<td>8 8 27.4 lin ✓</td>
<td>128 8 17.1 lin ✓</td>
</tr>
<tr>
<td>Haswell Xeon</td>
<td>2014</td>
<td>16 4 7.9 lin ✓</td>
<td>8 8 28.5 lin ✓</td>
<td>128 8 16.8 lin ✓</td>
</tr>
<tr>
<td>Skylake</td>
<td>2015</td>
<td>16 4 9.0 lin ✓</td>
<td>8 8 2.0 lin ✓</td>
<td>128 12 212.0 XOR-7 ✓</td>
</tr>
<tr>
<td>Broadwell Xeon</td>
<td>2016</td>
<td>16 4 8.0 lin ✓</td>
<td>8 8 18.2 lin ✓</td>
<td>256 6 272.4 XOR-8 ✓</td>
</tr>
<tr>
<td>Coffee Lake</td>
<td>2017</td>
<td>16 4 9.1 lin ✓</td>
<td>8 8 26.3 lin ✓</td>
<td>128 12 230.3 XOR-7 ✓</td>
</tr>
</tbody>
</table>
TLBLEED: TLB AS SHARED STATE
TLBLEED: TLB AS SHARED STATE

• Can we use only latency?

• Map many virtual addresses to same physical page
TLBLEED: TLB AS SHARED STATE

- Can we use only latency?
- Map many virtual addresses to same physical page
TLBLEED: TLB AS SHARED STATE
TLBLEED: TLB AS SHARED STATE

• Let’s observe EdDSA ECC key multiplication

```c
void _gcry_mpi_ec_mul_point (mpi_point_t result,
                            gcry_mpi_t scalar, mpi_point_t point,
                            mpi_ec_t ctx)
{
    ...
    for (j=nbits-1; j >= 0; j--) {
        _gcry_mpi_ec_dup_point (result, result, ctx);
        if (mpi_test_bit (scalar, j))
            _gcry_mpi_ec_add_points(result,result,point,ctx);
    }
    ...
}
```
TLBLEED: TLB AS SHARED STATE

- Let's observe EdDSA ECC key multiplication

- Scalar is secret and ADD only happens if there's a 1

```c
void _gcry_mpi_ec_mul_point (mpi_point_t result,
   gcry_mpi_t scalar, mpi_point_t point,
   mpi_ec_t ctx)
{
  ...
  for (j=nbits-1; j >= 0; j--) {
    _gcry_mpi_ec_dup_point (result, result, ctx);
    if (mpi_test_bit (scalar, j))
      _gcry_mpi_ec_add_points (result, result, point, ctx);
  }
  ...
}
```
Let's observe EdDSA ECC key multiplication

Scalar is secret and ADD only happens if there's a 1

Like RSA square-and-multiply

```c
void _gcry_mpi_ec_mul_point (mpi_point_t result, gcry_mpi_t scalar, mpi_point_t point, mpi_ec_t ctx)
{
...
for (j=nbits-1; j >= 0; j--) {
    _gcry_mpi_ec_dup_point (result, result, ctx);
    if (mpi_test_bit (scalar, j))
        _gcry_mpi_ec_add_points (result, result, point, ctx);
}
...}
```
TLB-BLEED: TLB AS SHARED STATE

- Let’s observe EdDSA ECC key multiplication
- Scalar is secret and ADD only happens if there’s a 1
- Like RSA square-and-multiply
- But: we can not use code information! Only data...

```c
void _gcry_mpi_ec_mul_point (mpi_point_t result,
gcry_mpi_t scalar, mpi_point_t point,
mpi_ec_t ctx)
{
...
    for (j=nbits-1; j >= 0; j--) {
        _gcry_mpi_ec_dup_point (result, result, ctx);
        if (mpi_test_bit (scalar, j))
            _gcry_mpi_ec_add_points (result, result, point, ctx);
    }
...
}
```
TLBLEED: TLB AS SHARED STATE
TLBLEED: TLB AS SHARED STATE

• Let’s find the spatial L1 DTLB separation

• There isn’t any

• Too much activity in both blue/green cases
TLBLEED: TLB AS SHARED STATE

- Let’s find the spatial L1 DTLB separation
- There isn’t any
- Too much activity in both blue/green cases
TLBLEED: TLB AS SHARED STATE

• Let's find the spatial L1 DTLB separation
• There isn't any
• Too much activity in both blue/green cases
TLBLEED: TLB AS SHARED STATE
TLBLEED: TLB AS SHARED STATE

- Monitor a single TLB set and use temporal information
TLBLEED: TLB AS SHARED STATE

• Monitor a single TLB set and use temporal information

• Use machine learning (SVM classifier) to tell the difference
TLBLEED: TLB AS SHARED STATE

- Monitor a single TLB set and use temporal information
- Use machine learning (SVM classifier) to tell the difference
TLB BLEED: TLB AS SHARED STATE

- Monitor a single TLB set and use temporal information
- Use machine learning (SVM classifier) to tell the difference
TLB BLEED: TLB AS SHARED STATE

- Monitor a single TLB set and use temporal information
- Use machine learning (SVM classifier) to tell the difference
TL BLEED: TLB AS SHARED STATE

- Monitor a single TLB set and use temporal information.
- Use machine learning (SVM classifier) to tell the difference.
TLBLEED: TLB AS SHARED STATE

• Monitor a single TLB set and use temporal information

• Use machine learning (SVM classifier) to tell the difference
TLBLEED RELIABILITY: ECC
## TLB Bledger Reliability: ECC

<table>
<thead>
<tr>
<th>Microarchitecture</th>
<th>Trials</th>
<th>Success</th>
<th>Median BF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skylake</td>
<td>500</td>
<td>0.998</td>
<td>$2^{1.6}$</td>
</tr>
<tr>
<td>Broadwell</td>
<td>500</td>
<td>0.982</td>
<td>$2^{3.0}$</td>
</tr>
<tr>
<td>Coffeelake</td>
<td>500</td>
<td>0.998</td>
<td>$2^{2.6}$</td>
</tr>
<tr>
<td>Total</td>
<td>1500</td>
<td>0.993</td>
<td></td>
</tr>
</tbody>
</table>
TLB BLEED RELIABILITY: ECC

- Single trace capture: 1ms
- Median end-to-end time: 17s

<table>
<thead>
<tr>
<th>Microarchitecture</th>
<th>Trials</th>
<th>Success</th>
<th>Median BF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skylake</td>
<td>500</td>
<td>0.998</td>
<td>$2^{1.6}$</td>
</tr>
<tr>
<td>Broadwell</td>
<td>500</td>
<td>0.982</td>
<td>$2^{3.0}$</td>
</tr>
<tr>
<td>Coffeelake</td>
<td>500</td>
<td>0.998</td>
<td>$2^{2.6}$</td>
</tr>
<tr>
<td>Total</td>
<td>1500</td>
<td>0.993</td>
<td></td>
</tr>
</tbody>
</table>
TLBLEED RELIABILITY: ECC

Single trace capture: 1ms
Median end-to-end time: 17s

<table>
<thead>
<tr>
<th>Microarchitecture</th>
<th>Trials</th>
<th>Success</th>
<th>Median BF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skylake</td>
<td>500</td>
<td>0.998</td>
<td>2^{1.6}</td>
</tr>
<tr>
<td>Broadwell</td>
<td>500</td>
<td>0.982</td>
<td>2^{3.0}</td>
</tr>
<tr>
<td>Coffeclipse</td>
<td>500</td>
<td>0.998</td>
<td>2^{2.6}</td>
</tr>
<tr>
<td>Total</td>
<td>1500</td>
<td>0.993</td>
<td></td>
</tr>
</tbody>
</table>
TLB BLEED RELIABILITY: ECC

- Single trace capture: 1ms
- Median end-to-end time: 17s

<table>
<thead>
<tr>
<th>Microarchitecture</th>
<th>Trials</th>
<th>Success</th>
<th>Median BF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skylake</td>
<td>500</td>
<td>0.998</td>
<td>2^{1.6}</td>
</tr>
<tr>
<td>Broadwell</td>
<td>500</td>
<td>0.982</td>
<td>2^{3.0}</td>
</tr>
<tr>
<td>Coffee Lake</td>
<td>500</td>
<td>0.998</td>
<td>2^{2.6}</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>1500</td>
<td>0.993</td>
<td></td>
</tr>
</tbody>
</table>
TLBLEED RELIABILITY: RSA

- 1024-bit RSA square-and-multiply in libgcrypt
- Old version
- F+R hardened: conditional pointer swap
TLBLEED RELIABILITY: RSA

- 1024-bit RSA square-and-multiply in libgcrypt
- Old version
- F+R hardened: conditional pointer swap
RECEPTION

- Intel: dismissed TLBleed
- OpenBSD disabled Intel HT
- Widespread media coverage, logo thanks to TheRegister
- Wikipedia
RECEPTION

- Intel: dismissed TLBleed

- OpenBSD disabled Intel HT

- Widespread media coverage, logo thanks to TheRegister

- Wikipedia
RECEPTION

- Intel: dismissed TLBleed
- OpenBSD disabled Intel HT
- Widespread media coverage, logo thanks to TheRegister
- Wikipedia
RECEPTION

- Intel: dismissed TLBleed
- OpenBSD disabled Intel HT
- Widespread media coverage, logo thanks to TheRegister
- Wikipedia
• Work also by Kaveh Razavi, Cristiano Giuffrida, Herbert Bos

• Some diagrams in these slides were taken from other work: FLUSH+RELOAD, Cloak

• Yuval Yarom, Katrina Falkner, Peter Peßl, Daniel Gruss
CONCLUSION

- Practical, reliable, high resolution side channels exist outside the cache.
CONCLUSION

- Practical, reliable, high resolution side channels exist outside the cache
- They bypass defenses
CONCLUSION

- Practical, reliable, high resolution side channels exist outside the cache
- They bypass defenses
- @bjg @kavehrazavi
CONCLUSION

• Practical, reliable, high resolution side channels exist outside the cache

• They bypass defenses

• @bjg @kavehrazavi

• @vu5ec
CONCLUSION

• Practical, reliable, high resolution side channels exist outside the cache
• They bypass defenses

• @bjg @kavehrazavi
• @vu5ec
• vusec.net/projects/tlbleed/
CONCLUSION

- Practical, reliable, high resolution side channels exist outside the cache
- They bypass defenses
- @bjg @kavehrazavi
- @vu5ec
- vusec.net/projects/tlbleed/
- Thank you for listening