Ultimate SLH: Taking Speculative Load Hardening to the Next Level

Zhiyuan Zhang, Gilles Barthe, Chitchanok Chuengsatiansup, Peter Schwabe, Yuval Yarom
Spectre-V1 Revisit

if (index < arrayLen) {
    x = array[index];
    y = array2[4096];
}

Obtain secret
Misprediction
Leak
Covert Channel
Speculative Load Hardening (SLH)

mask = 0;
if (index < arrayLen) {
    r -1 ≤ index < arrayLen ? mask : -1
    -1 = array[index] | r -1 ≤;
}

Fixed memory access under misprediction

Track speculative state

Poison memory
Limitation of SLH

• **Only** protect memory

• Leakages could also from *Control Flow (ccs 2021)*

<table>
<thead>
<tr>
<th></th>
<th>SLH</th>
<th>SSLH (ccs 2021)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Control Flow</td>
<td>X</td>
<td>✔</td>
</tr>
</tbody>
</table>
Leakage Model

• Constant-time Model
  • Memory
  • Control Flow
  • Variable-time Instructions

• Bring constant-time model to speculative execution

<table>
<thead>
<tr>
<th></th>
<th>SLH</th>
<th>SSLH (CCS 2021)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Control Flow</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Variable-time</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

Really Leak?
Timing variable-time instructions

```c
value = sqrtSD(value);
value = mulSD(value, value);
```

On i7-10710U:

- Executing a pair of SQRTSD and MULSD:
  - 65536: 5 cycles
  - 2.34e-308: 7 cycles
Timing speculative variable-time instructions

if (isPublic) {
    value = sqrtsgd (value);
    value = mulsd (value, value);
}
if (isPublic) {
    value = sqrttsd (value);
    value = mulsd (value, value);
    value = sqrttsd (value);
    value = mulsd (value, value);
    ....
}

independent_access(x);
Ultimate Speculative Load Hardening

• Constant-time Model
  • Memory access (load + store)
  • Control Flow Transfers
  • Variable-time Instructions

• Bring constant-time model to speculative execution

<table>
<thead>
<tr>
<th></th>
<th>SLH</th>
<th>SSLH (CCS 2021)</th>
<th>USLH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Control Flow</td>
<td>✗</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Variable-time</td>
<td>✗</td>
<td>✗</td>
<td>✅</td>
</tr>
</tbody>
</table>

Really Leak? Really Leak!
Ultimate Speculative Load Hardening

if (isPublic) {
    mask = isPublic ? Mask : -1;
    value = value | mask;
    value = sqrtsd (value);
    value = mulsd (value, value);
    ......  
    independent_access(x);
}

Compiled by SLH

Compiled by USLH
USLH Benchmark

Benchmark with SPEC2017

Time Cost Normalisation

- **14%** - int_rate
- **13%** - int_speed
- **43%** - fp_rate
- **45%** - fp_speed

Legend:
- No Mitigation
- SLH
- USLH
- No Speculation
Summary

• Leakages could from variable-time operations
• Implement and benchmark Ultimate SLH
• Gadget search tool
• Formal Proof

https://github.com/0xADE1A1DE/USLH