Security Verification of Virtual Memory Implementations with CheckMate

Naorin Hossain
Princeton University

PI: Caroline Trippel
Stanford University

Intel SCAP Workshop
September 29, 2020
CheckMate: early stage verification tool for detecting hardware vulnerabilities to exploits

- Evaluates microarchitectural susceptibility to known security exploit classes and synthesizes security litmus tests for susceptible designs

CheckMate

Use SAT solver to:
1. Synthesize candidate executions of all possible programs (up to synthesis bound)
2. Map candidate executions to \( \mu \)arch model
3. Search for exploit pattern in \( \mu \)hb graph for candidate execution

\( \mu \)arch model
Exploit pattern

Security litmus test

Flush+Reload
- cache side-channel attack pattern

Standard OoO processor

Prime+Probe
- cache side-channel attack pattern

Exploit
Speculative cache pollution

New attacks!
MeltdownPrime,
SpectrePrime

Exploit
Speculative cache line in validation

CheckMate
SandyBridge model: µarch model with address translation for security analysis of VM systems

<table>
<thead>
<tr>
<th>Features for security analysis</th>
<th>Prior OoO processor model</th>
<th>SandyBridge model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Supported operation types</strong></td>
<td>User-level operations:</td>
<td>User-level operations:</td>
</tr>
<tr>
<td></td>
<td>• Memory access</td>
<td>• Memory access</td>
</tr>
<tr>
<td></td>
<td>• Memory fence</td>
<td>• Memory fence</td>
</tr>
<tr>
<td></td>
<td>• Branch</td>
<td>System-level operations:</td>
</tr>
<tr>
<td></td>
<td>• Cache flush</td>
<td>• Remap V-to-P address mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TLB entry invalidation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hardware-level operations:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Page table walk</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Dirty bit update</td>
</tr>
<tr>
<td><strong>Structures accessible for exploits</strong></td>
<td>Cache</td>
<td>Cache, TLB</td>
</tr>
<tr>
<td><strong>Virtual memory assumptions</strong></td>
<td>One-to-one V-to-P address mapping (no synonyms)</td>
<td>Synonyms permitted</td>
</tr>
</tbody>
</table>

**SandyBridge stages**

- FetchDispatch
- IssueAGU
- AccessTLB
- AccessCache
- WriteBack
- Commit
- StoreBuffer
- ViTLCreate
- ViTLEXPire
- ViCLCreate
- ViCLEXPire
- MainMemory
- Complete
TLB flush+reload pattern synthesized on SandyBridge model

<table>
<thead>
<tr>
<th>Attacker T0 on C0</th>
<th>Victim T1 on C0</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i0) R [VA1] → r2</td>
<td>(i0) R [VA1] → r1</td>
</tr>
<tr>
<td>(i1) ptwalk [VA0] → TLB entry1</td>
<td>(i1) ptwalk [VA0] → TLB entry2</td>
</tr>
<tr>
<td>(i2) INVLPG [VA1]</td>
<td></td>
</tr>
</tbody>
</table>

Flush

Reload