Addendum to
RETBLEED: Arbitrary Speculative Code Execution with Return Instructions

Johannes Wikner
ETH Zurich
Daniël Trujillo
ETH Zurich
Kaveh Razavi
ETH Zurich

Abstract
Speculative execution attacks that exploit branch target injection (Spectre-BTI) have so far been limited to indirect branch instructions. RETBLEED extends Spectre-BTI to return instructions. During our research into RETBLEED, we discovered that certain AMD CPUs are also vulnerable to a new class of speculative execution attacks that we refer to as PHANTOM JMPs—incorrect branch target prediction in the absence of a corresponding branch instruction. Our investigation so far shows that PHANTOM JMPs significantly increases the attack surface of Spectre-BTI, but it is more difficult to exploit under real-world conditions. This addendum briefly discusses our findings and directions for future research.

1 PHANTOM JMPs
CPU pipelines embody instruction fetch units that provide input for instruction decode units. An instruction fetch loads a line from the instruction cache to be decoded. Only after decode and potentially execute of the line will the CPU frontend become aware of the definitive location of the next instruction fetch. This means that the frontend has to decide whether to branch or not before it has decoded the current line, before it knows whether it had just fetched a branch. It decides this based on information provided by the branch predictor.

By tricking the branch predictor to believe that there exists a branch at a certain location, we trigger a speculative branch, even when a corresponding branch instruction is nonexistent. This enables speculative code execution coming from arbitrary instructions, commonly known as phantom branches. We confirm that AMD Zen 1, Zen 1+ and Zen 2 CPUs are affected by PHANTOM JMPs, similar to RETBLEED.

2 Comparison with RETBLEED
We can force a speculative branch at arbitrary instructions using the same method as RETBLEED. However, we observe significantly shorter speculation windows when hijacking instructions other than returns. Mispredicted returns allow for multiple sequential memory loads in the speculation window, but for arbitrary instructions, such as nop or memory operations, we only observe a single speculative load.

This means that to enable leakage through a covert channel such as FLUSH+RELOAD, the secret data needs to already be available in a register before the phantom branch. We also verified that PHANTOM JMPs can leak secret data from memory loaded into a register under an already-ongoing speculative execution (e.g., Spectre-BCB). Further research is necessary to understand the attack surface of PHANTOM JMPs in the architectural and speculative scenarios for privileged software.

3 Mitigating PHANTOM JMPs
AMD’s mitigation removes branch predictions associated with returns in the kernel, which exclusively mitigates RETBLEED. To mitigate PHANTOM JMPs in the kernel, the only generic mitigation for this family of CPUs is flushing all branch predictions on every user–kernel transition using Indirect Branch Prediction Barriers (IPBPs). Flushing all branch target predictions, in lieu of flushing return predictions, incurs an overhead of 209% according to our benchmarks on an AMD Zen 1 system. Further research is necessary for reducing this overhead without sacrificing security.

4 Timeline and Further Analysis
We became aware of AMD’s decision to include PHANTOM JMPs as part of the RETBLEED’s disclosure shortly before the embargo date. While we discovered PHANTOM JMPs shortly after RETBLEED, we have only investigated it since early 2022 due to the required efforts around RETBLEED and its mitigation. We plan to publish a more comprehensive analysis of PHANTOM JMPs at a later date. Below is the timeline of our RETBLEED and PHANTOM JMPs research:

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 2021</td>
<td>Discovery of RETBLEED</td>
</tr>
<tr>
<td>Aug 2021</td>
<td>Discovery of PHANTOM JMPs</td>
</tr>
<tr>
<td>Feb 2022</td>
<td>Disclosure of RETBLEED to Intel and AMD</td>
</tr>
<tr>
<td>Jun 2022</td>
<td>Informed of AMD’s disclosure plan</td>
</tr>
<tr>
<td>Jul 2022</td>
<td>Public disclosure of both issues</td>
</tr>
</tbody>
</table>