Addendum 2 to
RIDL: Rogue In-flight Data Load

Stephan van Schaik*, Alyssa Milburn*, Sebastian Österlund*, Pietro Frigo*, Giorgi Maisuradze†‡,
Kaveh Razavi*, Herbert Bos*, and Cristiano Giuffrida*

*Department of Computer Science
Vrije Universiteit Amsterdam, The Netherlands
{s.j.r.van.schaik, a.a.milburn, s.osterlund, p.frigo}@vu.nl,
{kaveh, herbertb, giuffrida}@cs.vu.nl

†CISPA Helmholtz Center for Information Security
Saarland Informatics Campus
giorgi.maisuradze@cispa.saarland

***Abstract*—On Jan 27, 2020, we (VUSec) disclose two "new" RIDL/MDS variants at the end of another (third) embargo. We do not think either of these variants is particularly novel or interesting and they are simply "more RIDL" (focusing on ways to get data into microarchitectural buffers RIDL can leak from).**

A. Flawed MDS mitigations (encore!)

Intel's original microcode update, which modifies the VERW instruction to clear CPU buffers and mitigate MDS, was flawed in that it cleared the buffers using stale (potentially sensitive) data on several of the CPUs we used for testing (e.g., i7-7700K). This meant that data could still be leaked across privilege boundaries using RIDL even if SMT has been disabled and the recommended VERW mitigation has been applied. See our first addendum (on TAA and VERW-bypass) for more details.

Then in November 2019, Intel published another set of updates to mitigate the RIDL/MDS vulnerability once and for all. Unfortunately, they failed again, as the fix still leaves multiple avenues for exploitation. None of these issues is new. They merely support the claim in the original paper that there are multiple ways to and multiple buffers to leak from. One of the issues (L1DES) was insisted on by us ever since we first shared the RIDL pre-final paper in January 2019. The other issue is a one-line of code change from one of the PoCs we originally submitted.

B. L1D Eviction Sampling (L1DES)

The first issue, which Intel refers to as L1D Eviction Sampling (L1DES), is a RIDL variant that leaks from L1D evictions (assigned CVE-2020-0549). It may seem that sometimes history does repeat itself, because this is again something that we had already shown in our original RIDL paper, as shown in Figure 6. In the camera-ready version of the RIDL paper, we also explicitly mentioned that, at every context switch, "the entire L1 cache needs to be flushed first since the entries go through the LFBs" to properly mitigate RIDL. We removed this sentence in the original version of the RIDL paper released on May 14, 2019, since Intel had not yet mitigated the RIDL variant based on L1D evictions (which would eventually become L1DES). Since then, we spent months trying to convince Intel that leaks from L1D evictions were possible and needed to be addressed.

On Oct 25, 2019, we reported to Intel that this variant would bypass their latest VERW mitigation (and so did a PoC shared with Intel on May 10, 2019), resulting in Intel finally acknowledging the L1D eviction issue and requesting another (L1DES) embargo. We learned that Intel had not found this issue internally and that the only other independent finder was the Zombieload team, which reported a PoC to Intel in May, 2019. Our RIDL-L1DES PoC is available on GitHub[1].

C. Vector Register Sampling (VRS)

The second issue, which Intel refers to as Vector Register Sampling (VRS), is a RIDL variant that leaks from vector registers (assigned CVE-2020-0548). This variant shows that RIDL can also leak values that are never even stored in memory. In reality, this is possible with a small, 1-line of code variation of our 'alignment write' PoC (originally leaking values stored in memory using alignment faults), which we shared with Intel on May 11, 2019. Since then, we spent months trying to convince Intel that our 'alignment write' PoC and its variations needed to be properly addressed.

On Oct 1, 2019, we reported to Intel that a 1-line modification of our 'alignment write' PoC can leak vector register values, resulting in Intel requesting a new (VRS) embargo. We are not aware of other independent finders to acknowledge for VRS. Our RIDL-VRS PoC is available on GitHub[1].

D. Conclusion

This research—some of whose details were withheld from the public version of the RIDL paper due to responsible disclosure considerations—further supports the arguments presented in our original paper. We reiterate that RIDL-class vulnerabilities are non-trivial to fix or mitigate, and current "spot" mitigation strategies for resolving these issues are questionable. Moreover, we question the effectiveness of year-long disclosure processes and also raise concerns on their disruptive impact on the academic process. We continue to work with Intel to improve their coordinated disclosure process and collaboration with academia.

[1] https://github.com/vusec/ridl