clARMOR: A Dynamic Buffer Overflow Detector for OpenCL Kernels

Christopher Erb
AMD Research
Advanced Micro Devices, Inc.
Austin, Texas, USA
Christopher.Erb@amd.com

Joseph L. Greathouse
Radeon Technologies Group
Advanced Micro Devices, Inc.
Austin, Texas, USA
Joseph.Greathouse@amd.com

ABSTRACT

Buffer overflows are a common source of program crashes, data corruption, and security exploits. While many tools exist to find these issues in CPU programs, buffer overflows are also problematic in heterogeneous systems serviced by languages such as OpenCL™. Existing buffer overflow detectors for heterogeneous systems are either limited to particular vendors or require heavyweight instrumentation that results in large runtime overheads.

This work describes clARMOR, an open source buffer overflow detector for OpenCL kernels and APIs. clARMOR is vendor and device neutral; we demonstrate its operation on multiple device types from a variety of vendors. Rather than requiring heavyweight kernel instrumentation, clARMOR uses canary regions to quickly tell if data is written outside of any global memory buffer. Rather than analyzing every memory access, clARMOR instead verifies that the canary regions have not been modified after each user kernel finishes. In some cases, clARMOR uses the target device to check these canary regions, further reducing the overheads. We show experiments demonstrating that, across 143 open source OpenCL benchmarks, clARMOR causes an average slowdown of 9.6% while still finding multiple real kernel buffer overflows.

CCS CONCEPTS

• **Computer systems organization** → **Heterogeneous (hybrid) systems**; • **Software and its engineering** → **Software maintenance tools**; • **Security and privacy** → Software and application security;

KEYWORDS

Buffer Overflow Detection, OpenCL, GPGPU, Open Source Software

ACM Reference Format:
Christopher Erb and Joseph L. Greathouse. 2018. clARMOR: A Dynamic Buffer Overflow Detector for OpenCL Kernels. In *IWOCL '18: International Workshop on OpenCL (IWOCL '18), May 14–16, 2018, Oxford, United Kingdom.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3204919.3204934

1 INTRODUCTION

Buffer overflows are a well-acknowledged challenge in the CPU world, and many detection tools exist to help developers find them [1, 8, 12, 13]. Buffer overflows are also a challenge on heterogeneous accelerators, but there are markedly fewer tools for their detection.

Heterogeneous systems now share memory between hosts and devices, meaning that overflows from OpenCL™ devices may not only crash accelerator kernels, but also run the risk of corrupting host memory. These errors are particularly difficult to debug, since traditional tools, such as host-based watchpoints, may not even see the corruption happen. Recent research has demonstrated that accelerator buffer overflows can allow remote code execution on the device [2, 7], and this implies host-side code injection may also be possible. These challenges make a case for robust, fast overflow detection tools for heterogeneous accelerators.

Existing overflow detectors for accelerators have severe limitations. CUDA-MEMCHECK, for instance, only works on GPUs from Nvidia and on programs built using their proprietary CUDA language [5]. The WebCL Validator only works on kernels written in WebCL, which has seen little adoption [6]. Oclgrind includes a memory checking tool, and it works on OpenCL kernels, meaning that it is not vendor-limited and can work on a great deal of existing code [9]. Unfortunately, it presents itself to the kernels as a CPU device and then instruments the execution; this can cause it to run up to 300x slower than native execution, limiting its use on production code. GPU Ocelot's overheads present a similar challenge [4].

2 CLARMOR

This work presents clARMOR, an open source buffer overflow detector for OpenCL™ kernels. clARMOR requires no source modifications, because it uses the Linux® LD_PRELOAD facility to wrap OpenCL APIs [10]. In addition, it works on devices from multiple vendors; we have verified that it works on CPUs from AMD and Intel and GPUs from AMD and Nvidia.

clARMOR utilizes canary regions to quickly identify writes outside of OpenCL buffers caused by kernels and memory APIs. Buffer allocations APIs, such as `clCreateBuffer`, are caught and the buffer is extended to include extra guard values. clARMOR then catches kernel enqueues, and checkers verify that these guard values have not changed after the kernel completes. On some devices, such as GPUs, we asynchronously launch these checkers to the same device as the original kernel to decrease performance overheads [3].

Across 143 benchmarks, we observed an average performance overhead of 9.7%, as shown in Figures 1 and 2. We used 4KB canary regions, which caused an average memory overhead of 9.6%.

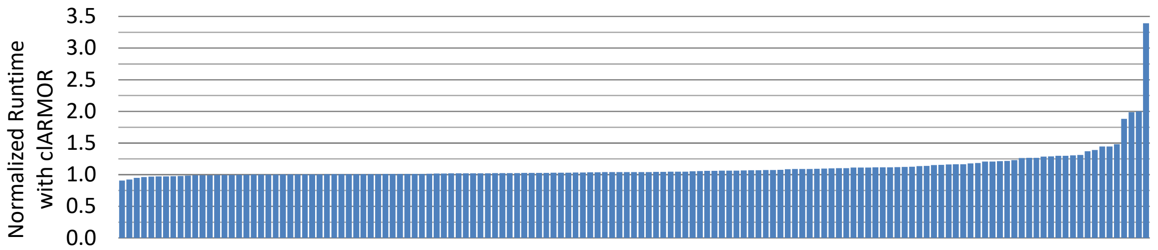

Figure 1: The runtime of 143 OpenCL™ benchmarks when using clARMOR normalized to their runtime without clARMOR. We tested using an AMD Ryzen™ 7 1800X CPU and AMD Radeon™ Vega Frontier Edition GPU running ROCm 1.7. The geometric mean of the slowdown caused by clARMOR was 9.6%, while the worst program (UnsharpMark from the AMD APP SDK samples) ran 2.39× slower because it had many short dependent kernels that exposed the overhead of clARMOR's checkers.

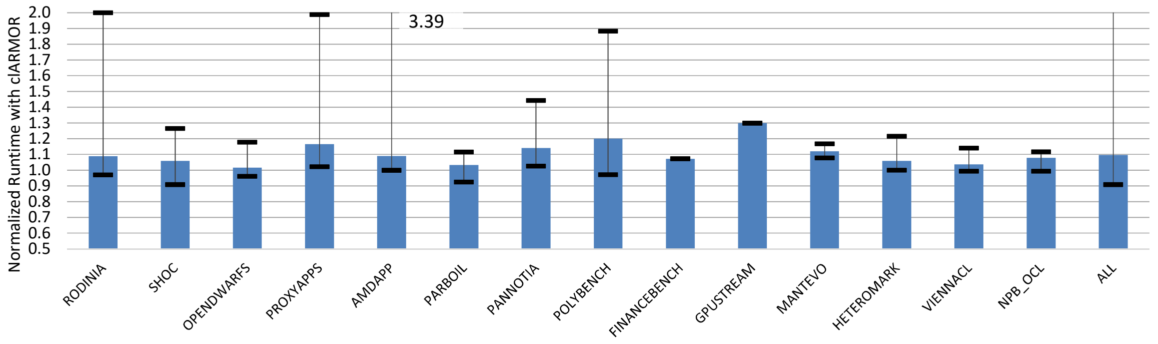

Figure 2: Normalized clARMOR execution time across various benchmark suites. Each bar shows the geometric mean (blue bar), minimum (bottom whisker), and maximum (top whisker) slowdown across the benchmarks within each suite.

clARMOR can detect overflows in global memory buffers, sub-buffers, OpenCL images, and both coarse- and fine-grained shared virtual memory regions. Since our initial academic publication about clARMOR [3], we have added support for detecting underflows (or, more precisely, overflows with negative indices) on all non-image buffer types. In addition, we added support for Intel's OpenCL runtime with CPU targets and AMD's ROCm platform.

We previously demonstrated that clARMOR found real buffer overflows in numerous open source OpenCL programs and benchmarks [3], many of which were fixed by their developers. Since then, we have used clARMOR to find that the CG benchmark in the SNU OpenCL version of the NAS Parallel Benchmarks has a number of data races that result in non-deterministic buffer underflows [11], and we have found and corrected multiple errors in proprietary programs.

clARMOR is available under an MIT license and can be found on GitHub at https://github.com/ROCm-Developer-Tools/clARMOR.

ACKNOWLEDGMENTS

AMD, the AMD Arrow logo, AMD Ryzen, Radeon, and combinations thereof are trademarks of Advanced Micro Devices, Inc. OpenCL is a trademark of Apple, Inc. used by permission by Khronos. Linux is a registered trademark of Linus Torvalds. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

REFERENCES

[1] Crispan Cowan, Calton Pu, Dave Maier, Jonathan Walpole, Peat Bakke, Steve Beattie, Aaron Grier, Perry Wagle, and Qian Zhang. 1998. StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks. In *Proc. of the USENIX Security Symposium*.

[2] Bang Di, Jianhua Sun, and Hao Chen. 2016. A Study of Overflow Vulnerabilities on GPUs. In *Proc. of the Int'l Conf. on Network and Parallel Computing (NPC)*.

[3] Chrisotpher Erb, Mike Collins, and Joseph L. Greathouse. 2017. Dynamic Buffer Overflow Detection for GPGPUs. In *Proc. of the Int'l Symp. on Code Generation and Optimization (CGO)*.

[4] Naila Farooqui, Andrew Kerr, Gregory Diamos, S. Yalamanchili, and K. Schwan. 2011. A Framework for Dynamically Instrumenting GPU Compute Applications within GPU Ocelot. In *Proc. of the Workshop on General-Purpose Computation on Graphics Processing Units (GPGPU)*.

[5] Geoff Gerfin and Vyas Venkataraman. 2012. Debugging Experience with CUDA-GDB and CUDA-MEMCHECK. Presented at GPU Technology Conference (GTC). (2012).

[6] Khronos Group. 2014. WebCL Validator. https://github.com/KhronosGroup/webcl-validator. (2014). Accessed: 2016-12-02.

[7] Andrea Miele. 2016. Buffer Overflow Vulnerabilities in CUDA: A Preliminary Analysis. *Journal of Computer Virology and Hacking Techniques* 12, 2 (May 2016), 113–120.

[8] Bruce Perens. 1987. Electric Fence. http://linux.die.net/man/3/efence. (1987). Accessed: 2016-12-02.

[9] James Price and Simon McIntosh-Smith. 2015. Oclgrind: An Extensible OpenCL Device Simulator. In *Proc. of the Int'l Workshop on OpenCL (IWOCL)*.

[10] Kevin Pulo. 2009. Fun with LD_PRELOAD. Presented at linux.conf.au. (2009).

[11] Sangmin Seo, Gangwon Jo, and Jaejin Lee. 2011. Performance Characterization of the NAS Parallel Benchmarks in OpenCL. In *Proc. of the IEEE Int'l Symp. on Workload Characterization (IISWC)*.

[12] Konstantin Serebryany, Derek Bruening, Alexander Potapenko, and Dmitriy Vyukov. 2012. AddressSanitizer: A Fast Address Sanity Checker. In *Proc. of the USENIX Annual Technical Conference (USENIX ATC)*.

[13] Julian Seward and Nicholas Nethercote. 2005. Using Valgrind to Detect Undefined Value Errors with Bit-Precision. In *Proc. of the USENIX Annual Technical Conference (USENIX ATC)*.