GuaNary: Efficient Buffer Overflow Detection in Virtualized Clouds Using Intel EPT-based Sub-Page Write Protection Support

Stella Bitchebe[*]
McGill University
Canada
stella.bitchebe@mcgill.ca

Yves KONE
Toulouse INP
France
yves.kone@ens-lyon.fr

Pierre Olivier
University of Manchester
United Kingdom
pierre.olivier@manchester.ac.uk

Jalil Boukhobza
ENSTA Bretagne
France
jalil.boukhobza@ensta-bretagne.fr

Yerom-David Bomberg
University of Rennes
France
david.bromberg@irisa.fr

Daniel Hagimont
Toulouse INP
France
Daniel.Hagimont@irit.fr

Alain Tchana
Grenoble INP
France
alain.tchana@grenoble-inp.fr

ABSTRACT

Buffer overflow is a widespread memory safety violation in C/C++, reported as the top vulnerability in 2022. Secure memory allocators are generally used to protect systems against attacks that may exploit buffer overflows. Existing allocators mainly rely on two types of countermeasures to prevent or detect overflows: canaries and guard pages, each with pros and cons in terms of detection latency and memory footprint.

This paper follows the Out of Hypervisor (OoH) trend for virtualized cloud applications. It introduces GuaNary, a novel safety guard against overflows allowing synchronous detection at a low memory footprint cost. OoH is a new virtualization research axis introduced in 2022 advocating the exposure of hardware features for virtualization to the guest OS so that its processes can take advantage of them. Based on the OoH principle, GuaNary leverages Intel Sub-Page write Permission (SPP), a recent hardware virtualization feature that allows to write-protect guest memory at the granularity of 128B (namely, sub-page) instead of 4KB. We implement a software stack, LeanGuard, which promotes the utilization of SPP from inside virtual machines by new secure allocators that use GuaNary. Our evaluation shows that for the same number of protected buffers, LeanGuardconsumes 8.3× less memory than SlimGuard, a state-of-the-art secure allocator. Furthermore, for the same memory consumption, LeanGuard protecting 25× more buffers than SlimGuard.

[*]This work was done when the author was still a Ph.D. student at Université Côte d'Azur in France.

ACM ISBN 979-8-4007-0624-0/24/06.
https://doi.org/10.1145/3652963.3655056

ACM Reference Format:
Stella Bitchebe, Yves KONE, Pierre Olivier, Jalil Boukhobza, Yerom-David Bomberg, Daniel Hagimont, and Alain Tchana. 2024. GuaNary: Efficient Buffer Overflow Detection in Virtualized Clouds Using Intel EPT-based Sub-Page Write Protection Support. In *Abstracts of the 2024 ACM SIGMETRICS/IFIP PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS/PERFORMANCE Abstracts '24), June 10–14, 2024, Venice, Italy.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3652963.3655056

1 BACKGROUND AND MOTIVATION

Allocators targetting buffer overflow mitigation must answer an important question: *How to detect and prevent an overflow?* Two common techniques have been studied to answer this question: canaries and guard pages[5]. Canaries are small 1-byte magic values located after a buffer and checked to detect overflow. They have a modest memory overhead but can only detect overflows asynchronously, i.e. when the value is checked. Guard pages are unmapped pages in the virtual address space, located after a buffer. Overflowing the buffer will trigger a fault if the page is hit.

Guard pages offer better security guarantees vs. canaries, as they prevent overflows through synchronous detection but at the cost of significant memory consumption. We measured up to 80× memory overhead, with the SlimGuard allocator[4], for the PARSEC-freqmine application with guard pages.

Only a few allocators have been developed with the primary goal of reducing memory overhead while preserving other important properties such as security and performance. Some allocators, such as OpenBSD[5], Cling[2], and DieHarder[6], attempted to reduce the memory footprint of linked list-based metadata by using bitmaps. However, they lead to significant performance degradation when the allocator performs randomization, a popular security guarantee technique. Hardware solutions have also been introduced to address buffer overflow. We can underline CHERI [7], which doubles the pointer size to include the bounds to the pointed buffer. This way, the hardware can check bounds violations. Such hardware solutions overcome buffer overflow. However, they include several limitations, mainly related to performance degradation, unpredictability, and the need to rewrite applications.

Synchronous Detection vs Memory Overhead: The Dilemma. Let us consider buf a vulnerable buffer. We define the *security distance of buf* as the number of bytes that separate it from a guard page. A zero security distance allows to immediately catch overflow attempts. The protection of all application buffers with a zero security distance is not practical for most existing allocators, as it would result in considerable memory overhead.

To cope with this problem, allocator designers combine guard pages with canaries and allow users to configure the security distance they desire according to their performance or memory budget. A protection frequency N places a guard page for every N-allocated buffers in each class, and canaries are used between buffers that are not at the boundary of a guard page. Fig. 1 presents, for different protection frequency values, the memory overhead and the security distance for the PARSEC-`blackscholes` application. As one might have imagined, the lower the frequency, the more memory waste and better the protection.

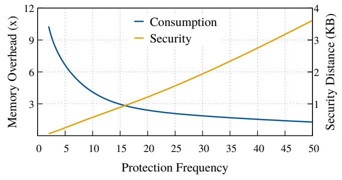

Figure 1: Memory overhead and average security distances for PARSEC-`blackscholes` when varying the protection frequency from 2 to 50.

2 CONTRIBUTIONS

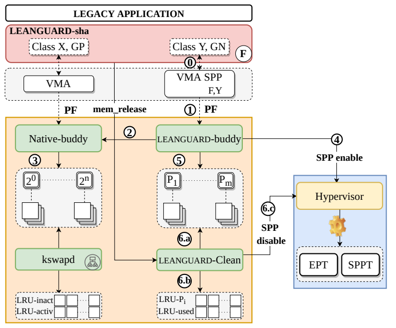

Figure 2: Architecture of LeanGuard.

(1) We introduce GuaNary, a novel type of safety guard for virtualized cloud-based applications. GuaNary leverages a recent Intel hardware virtualization feature called Sub-Page Write Permission [1] (SPP). SPP reduces write-protection granularity to 128B (called a sub-page) instead of 4KB. SPP was initially introduced to help hypervisors accelerate virtual machine's (VM) live migration/checkpointing. In this paper, following the OoH's (Out of Hypervisor [3]) mind, we repurpose SPP for security and make it exploitable by unprivileged VMs without breaking isolation between them. **(2)** We design LeanGuard, a system that exemplifies GuaNary in popular system software stacks. Fig. 2 depicts the architecture of LeanGuard bringing out all the software components and interactions such as the user application, the operating system, and the hypervisor. **(3)** We thoroughly evaluate LeanGuardusing micro- and macro-benchmarks (PARSEC applications), demonstrating its benefits.

3 KEY RESULTS

The main goal of LeanGuardis minimizing memory overhead while allowing synchronous buffer overflow detection. To this end, we compare LeanGuardwith different configurations of SlimGuard: SlimGuardwith only canaries (SlimG+Canary), SlimGuardwith only guard pages (SlimG+GP), and SlimGuardwith only GuaNaries(SlimG+GuaNary). SlimG+Canary is theoretically the lower bound regarding memory overhead, but it does not allow synchronous detection. So, from the perspective of memory overhead, the closer to SlimG+Canary, the better.

We run the PARSEC applications under each configuration while varying the protection frequency F. As we can see in Fig. 3 presenting the results, (1) LeanGuard leads almost to the same memory consumption as SlimG+Canary for these specific applications. (2) Our new safety guard GuaNary effectively leads to memory consumption reduction, as SlimG+GuaNary results suggest. (3) To protect a given proportion of buffers, SlimG+GP incurs a significant memory overhead compared to LeanGuard. For example, to protect 50% of the allocated buffers (F=2), LeanGuard, on average, uses 60% less memory than SlimG+GP. Using the same amount of memory as SlimG+GP, LeanGuard allows protecting 25× more buffers than SlimG+GP.

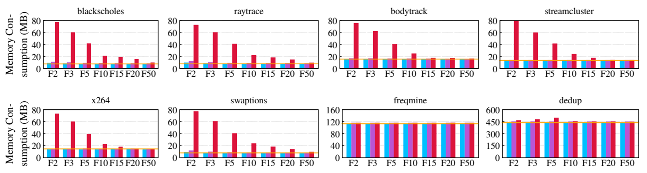

Figure 3: Memory consumption of each allocator configuration for PARSEC applications while varying the protection frequency.

REFERENCES

[1] 2017. Intel EPT-Based Sub-page Write Protection Support. https://lwn.net/Articles/736322/.
[2] Akritidis and Periklis. 2010. Cling: A Memory Allocator to Mitigate Dangling Pointers. 177–192.
[3] Stella Bitchebe and Alain Tchana. 2022. Out of Hypervisor (OoH): Efficient Dirty Page Tracking in Userspace Using Hardware Virtualization Features. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (Dallas, Texas) *(SC '22)*. IEEE Press, Article 87, 14 pages.
[4] Beichen Liu, Pierre Olivier, and Binoy Ravindran. 2019. SlimGuard: A Secure and Memory-Efficient Heap Allocator. In *Proceedings of the 20th International Middleware Conference* (Davis, CA, USA) *(Middleware '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3361525.3361532
[5] Otto Moerbeek. 2009. A new malloc (3) for OpenBSD. In *Proceedings of the 2009 European BSD Conference*, Vol. 9.
[6] Gene Novark and Emery D. Berger. 2010. DieHarder: Securing the Heap. In *Proceedings of the 17th ACM Conference on Computer and Communications Security* (Chicago, Illinois, USA) *(CCS '10)*. Association for Computing Machinery, New York, NY, USA, 573–584. https://doi.org/10.1145/1866307.1866371
[7] Jonathan Woodruff, Robert N.M. Watson, David Chisnall, Simon W. Moore, Jonathan Anderson, Brooks Davis, Ben Laurie, Peter G. Neumann, Robert Norton, and Michael Roe. 2014. The CHERI Capability Model: Revisiting RISC in an Age of Risk. *SIGARCH Comput. Archit. News* 42, 3 (june 2014), 457–468. https://doi.org/10.1145/2678373.2665740