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

Stella Bitchebe[1], Yves Kone[2], Pierre Olivier[3], Jalil Boukhobza[4], Yérom-David Bromberg[5], Daniel Hagimont[2], Alain Tchana[6]
[1]Université Côte d'Azur, [2]University of Toulouse, [3]University of Manchester, [4]ENSTA Bretagne, [5]University of Rennes, [6]Grenoble INP

1 Background and Motivation

For decades, the widespread usage of memory-unsafe languages like C and C++ raised the threat of security-related memory corruption errors representing vulnerabilities that attackers can exploit to execute malicious code, tamper with, and/or leak critical data. Google developers recently revealed that 70% of Google Chrome's bugs are related to memory management. Microsoft also made a similar observation. This paper focuses on buffer overflow, ranked the top vulnerability in 2022 by SANS Institute.

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. 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, for the PARSEC-freqmine application with guard pages.

Only a few allocators (or even improvements of existing allocators) have been developed with the primary goal of reducing the memory overhead while preserving other important properties such as security and performance. Some allocators, such as OpenBSD, Cling, and DieHarder, 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 (the most recent one), which doubles 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 (which limits their adoption).

2 Contributions

In this paper, (1) we introduce GuaNary, a novel type of safety guard for virtualized cloud-based applications. GuaNary provides the same security guarantee as guard pages against write overflows while drastically reducing memory overhead and with negligible performance overhead. GuaNary leverages a recent Intel hardware virtualization feature called Sub-Page Write Permission (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, we repurpose SPP for security and make it exploitable by unprivileged VMs without breaking isolation between them. (2) We design LeanGuard (Figure 1), a system that exemplifies GuaNary in popular system software stacks. (3) We thoroughly evaluate LeanGuard using micro- and macro-benchmarks (PARSEC applications), demonstrating its benefits.

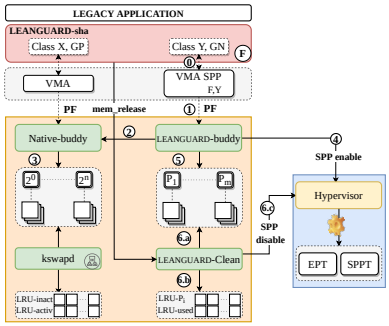

Figure 1. Architecture of LeanGuard.

3 Key Results

Our evaluation results show that LeanGuard, with the same memory consumption, can protect 25× more buffers compared to SlimGuard (that has already proven more efficient than recent state-of-the-art secure allocators). Inversely, to protect the same amount of buffer as SlimGuard, GuaNary requires about 8.3× less memory.

4 Main Artifact

We build LeanGuard by modestly extending the Xen hypervisor, the Linux kernel, and the SlimGuard secure allocator. The source of LeanGuard and all the artifacts are publicly available at https://github.com/bstellaceleste/OoH/tree/SPML/OoH-SPP.