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.

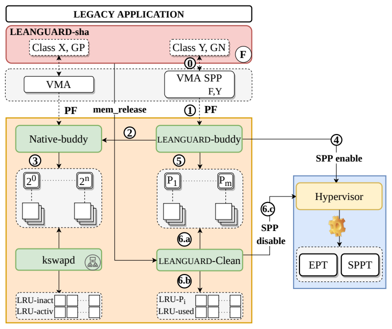

Figure 1. Architecture of LeanGuard.

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.

GuaNary: 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

EURO/SYS'23

1. Context and Motivation

- Buffer overflow was reported as the top vulnerability in 2022, according to the CWE (Common Weakness Enumeration) [1].
- Secure Allocators (e.g., Slimguard [2], Guarder [3], etc.) generally use *safety guards* located after a buffer to prevent and detect overflows.
- State-of-the-art safety guards:
  - **Canary**: 1-byte magic values checked to detect overflow => Modest memory overhead + Asynchronous detection (see Figure 2).
  - **Guard Page**: unmapped pages in the virtual address space that trigger a fault if the page is hit by an overflow => Significant memory overhead + Synchronous detection (see Figures 2 and 1).

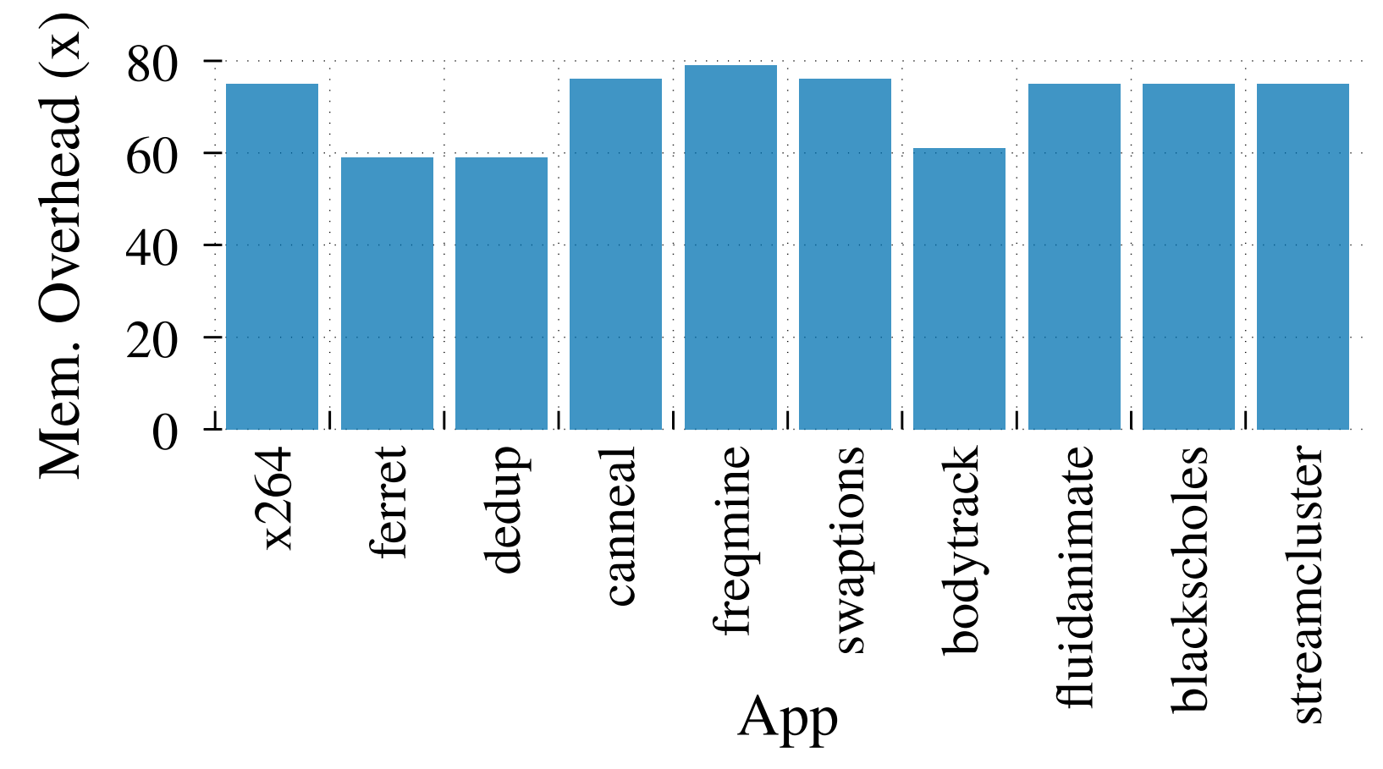

Figure 1: Memory over-consumption that Slimguard would incur for PARSEC applications if all the allocated buffers are placed at the boundary of a guard page (worse case).

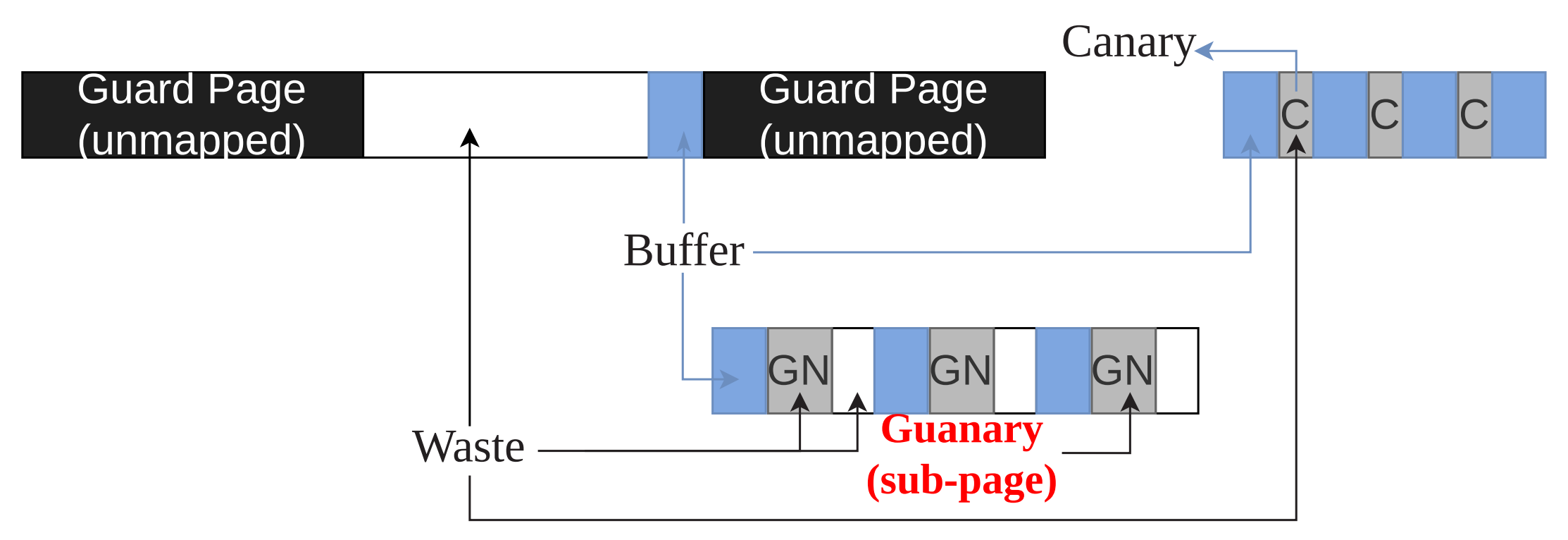

Figure 2: Canary, Guard pages, and GuaNary illustration. For the two latter, buffers are aligned with the lower boundary of the (sub)page.

2. Dilemma: Synchronuous Dectection vs. Memory Overhead

- Security distance: for a vulnerable buffer `b`, the security distance is the number of bytes separating `b` from a safety guard. A zero security distance allows catching overflow attempts immediately. Protecting all the buffers with a zero security distance is not practical for most existing allocators, as it would result in considerable memory overhead (like in Figure 1).
- Protection frequency: `F` is called the protection frequency if a safety guard is placed after every `F`-allocated buffers.
- Memory overhead is a real conundrum for users who sacrifice security for better memory utilization or vice versa. To this end, they can configure `F` for better memory consumption and security trade-off (See Figure 3).

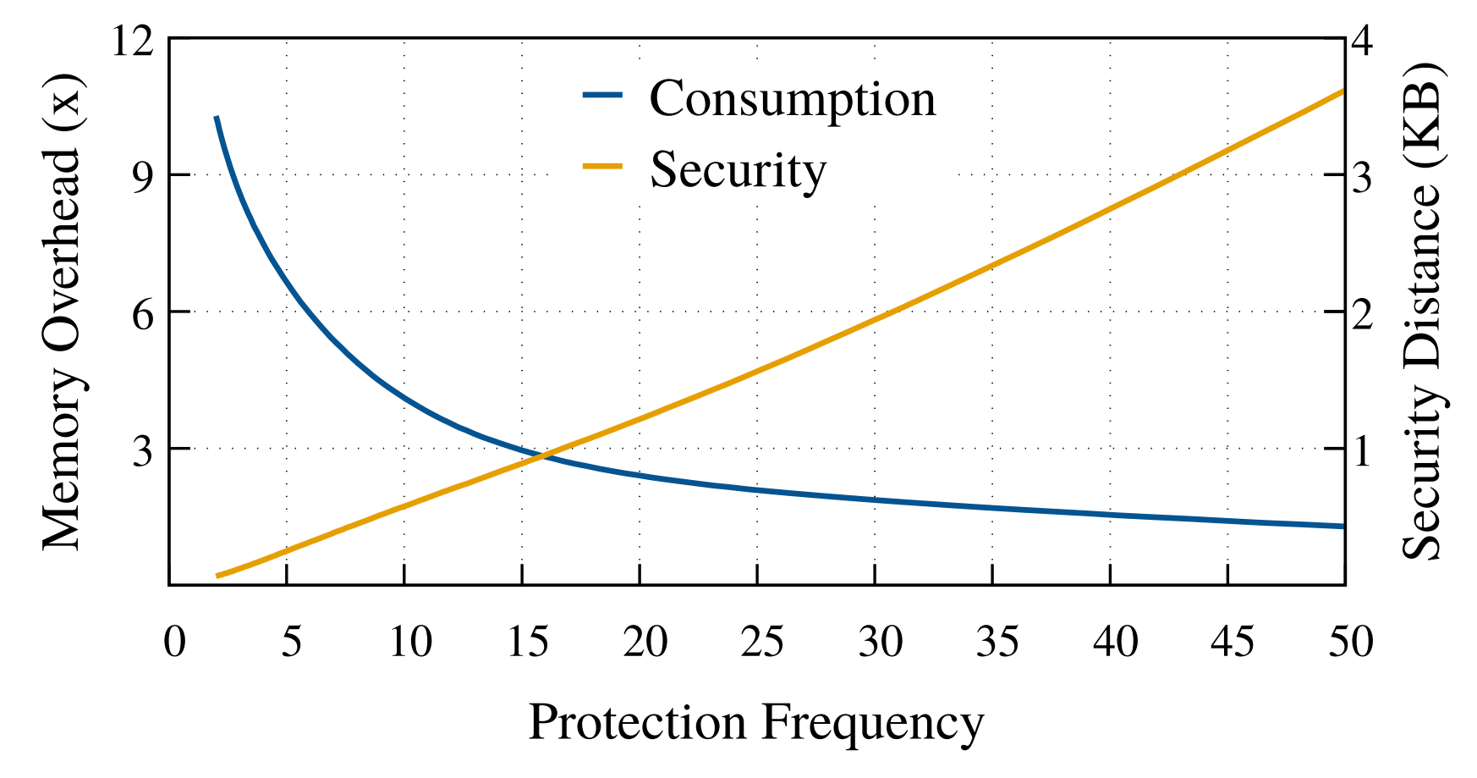

Figure 3: Memory overhead and average security distances for PARSEC-`blackscholes` when varying the protection frequency from 2 to 50. The intersection between the two curves gives the optimal frequency, i.e., the one providing the best memory overhead and security trade-off. The allocator is Slimguard.

3. Intel SPP: Sub-Page Write Permission

SPP [4] is a recent Intel hardware virtualization feature that allows the hypervisor to write-protect guest's memory at a sub-page (128B) granularity instead of 4KB (see Figure 6). SPP builds on top of the Extended Page Table (EPT) [5], introduced long ago to facilitate memory virtualization.

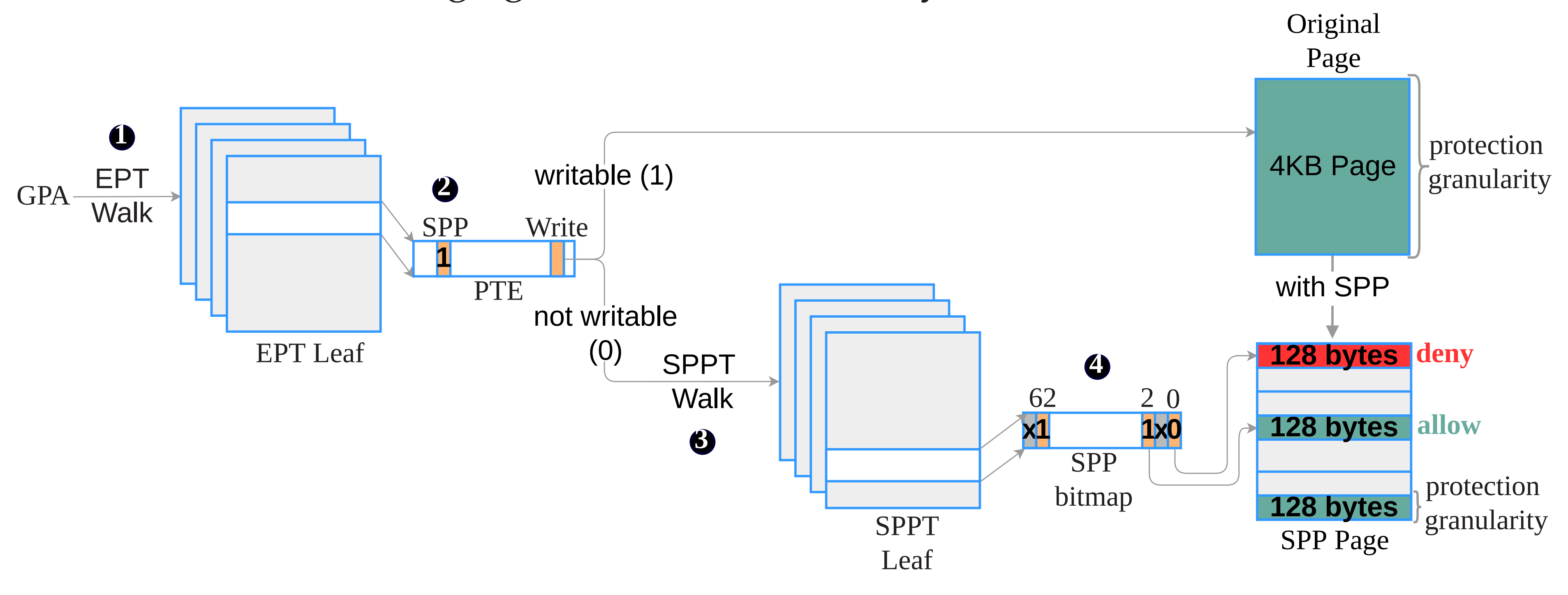

Figure 6: Overview of SPP functioning.

4. GuarNary and LeanGuard

Using SPP, we introduce **`GuarNary`**, a novel type of safety guard that is midway between `Guard` page and `caNary`, thus providing the advantages of both solutions: synchronous buffer overflow detection and modest memory consumption (see Figure 2).

We also propose **`LeanGuard`** (see Figure 4), a software stack for `GuarNary` usage from inside virtual machines by new secure allocators.

Figure 5 shows that for the same number of protected buffers, LeanGuard consumes 8.3× less memory than SlimGuard. Further, for the same memory consumption, LeanGuard allows protecting 25× more buffers than SlimGuard.

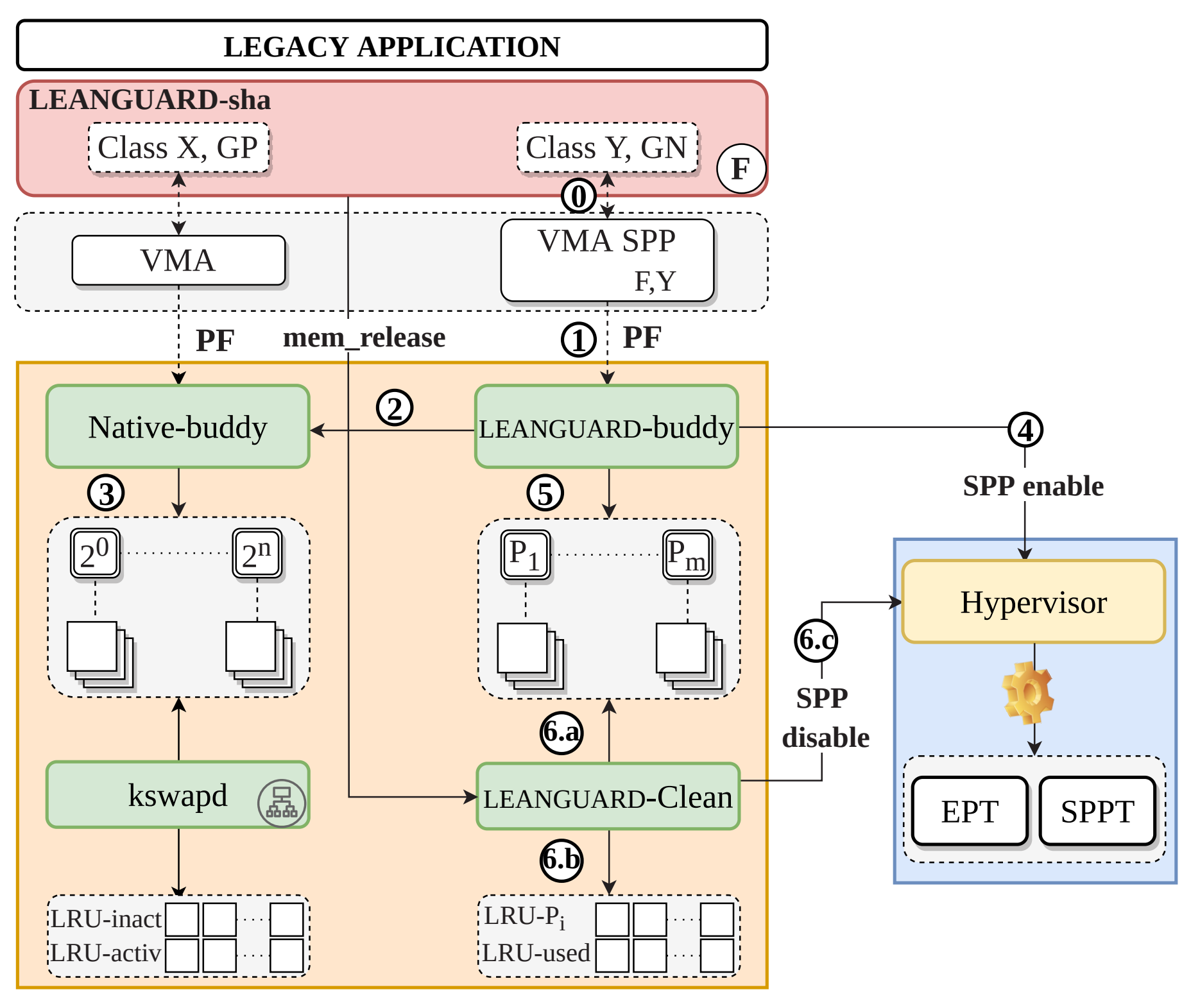

Figure 4: Architecture of LeanGuard.

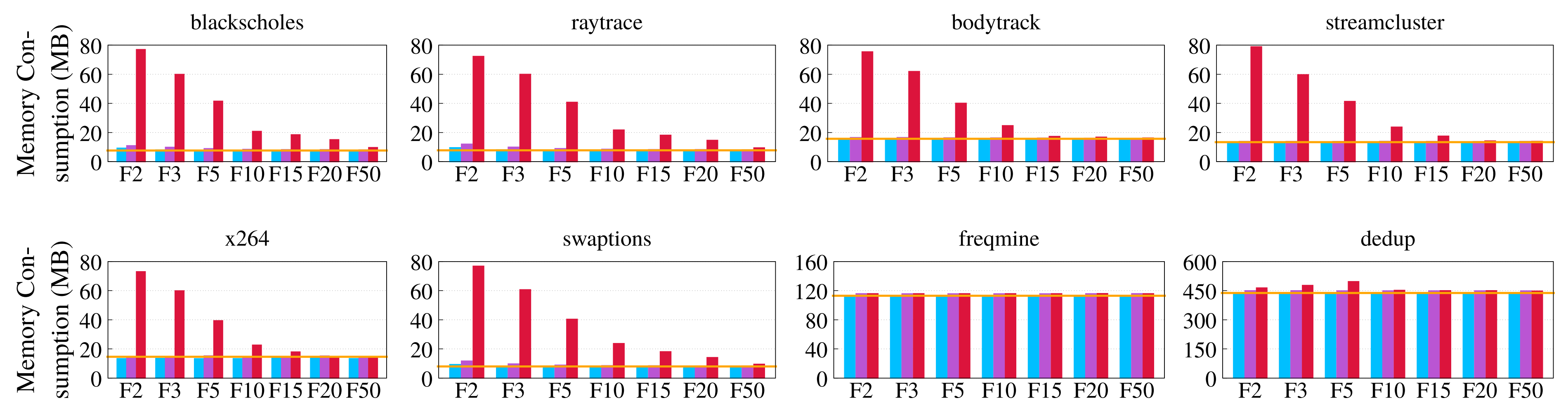

Figure 5: Memory consumption of each allocator configuration for PARSEC applications while varying the protection frequency.

References

[1] Cwe/sans top 25 most dangerous software errors. https://www.sans.org/top25-software-errors, 2022.

[2] Beichen Liu et al. Slimguard: A secure and memory-efficient heap allocator. *Middleware*, 2019.

[3] Sam Silvestro et al. Guarder: A tunable secure allocator. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 117–133, Baltimore, MD, August 2018. USENIX Association.

[4] Intel ept-based sub-page write protection support. https://lwn.net/Articles/736322/, Oct 2017.

[5] Intel. volume 3C. 2022.

Contact

bitchebe@i3s.unice.fr

alain.tchana@grenoble-inp.fr