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

EUROSYS'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).

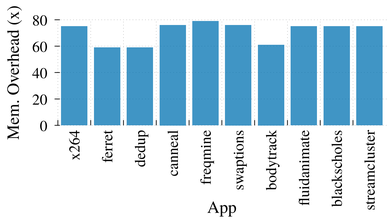

Figure 1: Memory over-consumption that Slimguard would incur for PARSEC applications if all the allocated buffers are placed at the boundary of a guard page (worse case).

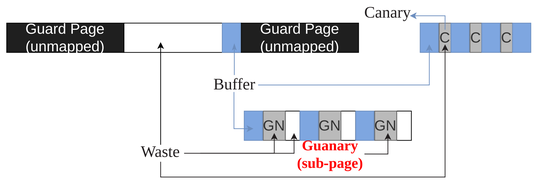

Figure 2: Canary, Guard pages, and GuaNary illustration. For the two latter, buffers are aligned with the lower boundary of the (sub)page.

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).

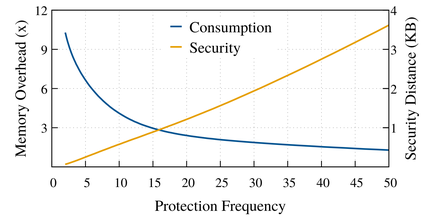

Figure 3: Memory overhead and average security distances for PARSEC-`blackscholes` when varying the protection frequency from 2 to 50. The intersection between the two curves gives the optimal frequency, i.e., the one providing the best memory overhead and security trade-off. The allocator is Slimguard.

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.

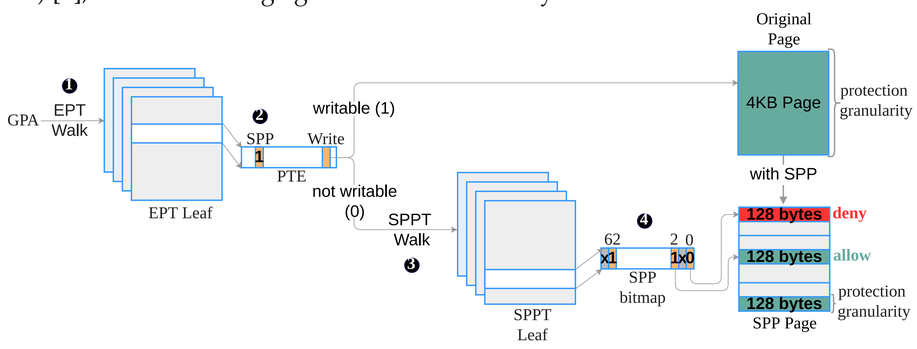

Figure 6: Overview of SPP functioning.

4. GuaNary and LeanGuard

Using SPP, we introduce **`GuarNary`**, a novel type of safety guard that is midway between `Guard` page and ca`Nary`, 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.

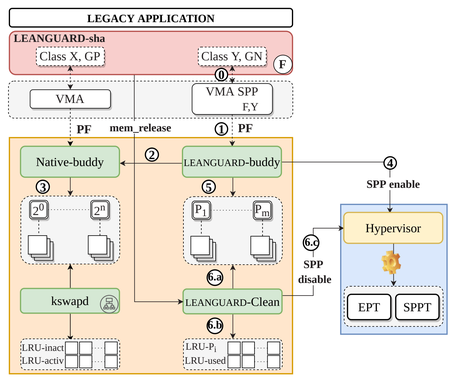

Figure 4: Architecture of LeanGuard.

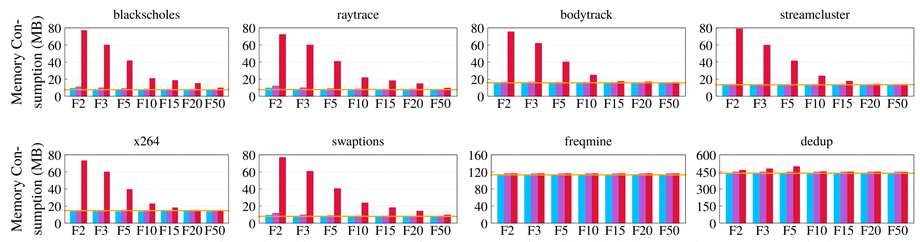

Figure 5: Memory consumption of each allocator configuration for PARSEC applications while varying the protection frequency.

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