Leveraging ARM TBI for Next-Generation Kernel Memory Sanitization

Yixuan Yang, Sandeep Tamrakar, Daniele Antonioli

What is TBI

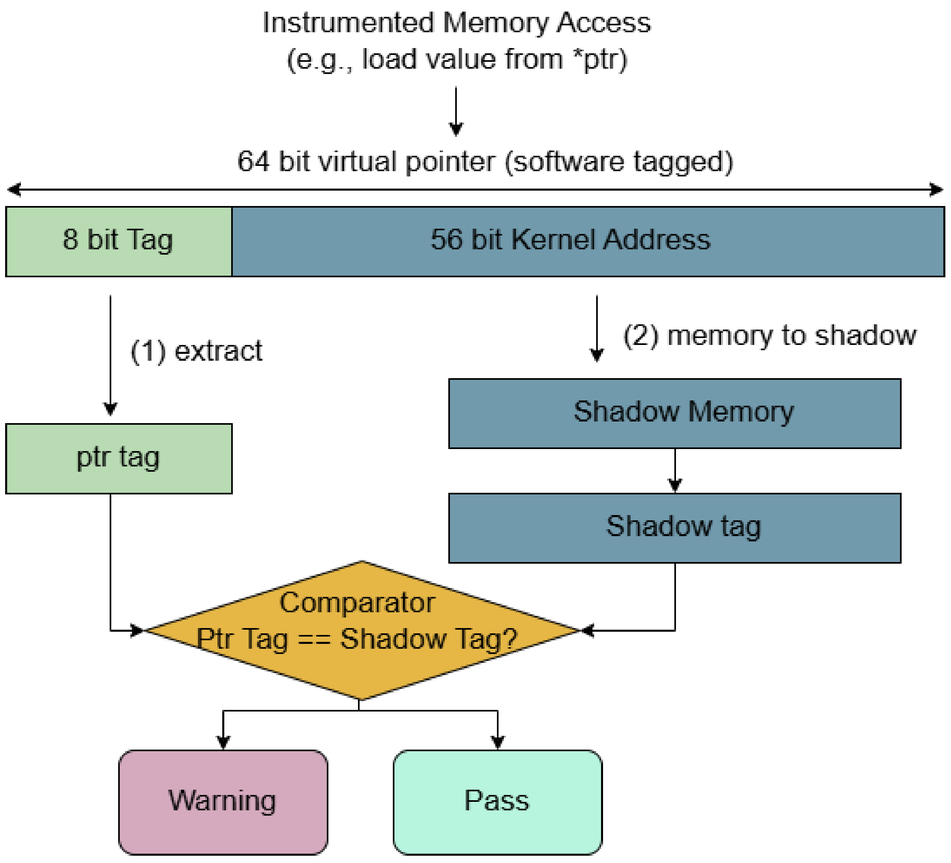
- Top Byte Ignore (TBI) is an ARM64 feature that allows the processor to ignore the top byte of a virtual address during memory access.

Motivation

- **Limitation in KMSAN**: KMSAN incurs significant memory overhead, which can be problematic in large-scale systems. To prevent memory exhaustion, the detection scope often has to be reduced, limiting its ability to identify uninitialized memory.
- **Potential of TBI**: TBI introduces a dedicated byte in memory addresses for tagging, enabling tags to represent states, metadata, or ownership information.

TBI current usage in Sanitizer

- **Software tag-based KASAN**: stores a tag in shadow memory and embeds the same tag into the pointer's top byte. On every memory access, instrumented code compares the pointer's tag with the shadow memory's tag.
- **MTSan:** utilizes Memory Tagging Extension to provide tag-based memory safety sanitizer with minimal runtime cost.



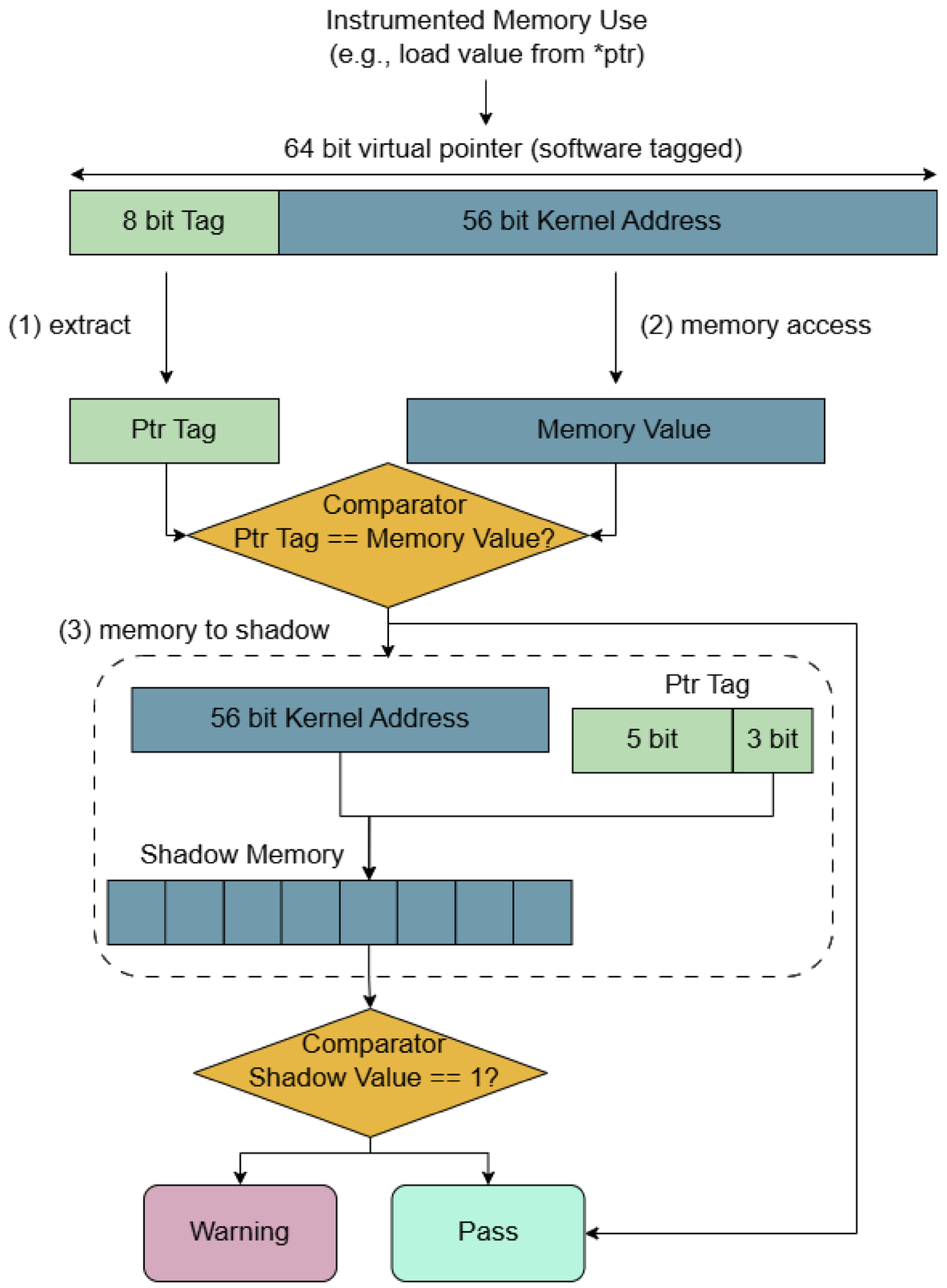
1. TBI implementation in address sanitizer

Lessons from Software tag-based KASAN

- **Tag as Metadata**: A tag acts as metadata denoting memory ownership, decoupling safety checks from the primary data.
- **Shadow Memory Principle**: Safety is verified by mapping the target address to a shadow that stores the expected tag.
- **Performance Benefit**: This architectural separation enables precise bug detection with significantly lower memory overhead compared to full redundancy.

Proposed TBI-enhanced KMSAN

- **Tag-as-Offset for Compact Shadow Memory**: We encode initialization state (1 bit per byte) in shadow memory, using the pointer tag's low 3 bits as an offset for lookup.
- **Tag Pre-Check for Selective Validation**: Before shadow memory access, we compare the pointer's tag against the value in the target memory. A tag mismatch serves as a fast path, confirming initialization and avoiding the slower shadow check for most memory accesses.



2. TBI-optimized KMSAN

Contribution

- **Expanded Tag Usage**: (1) as an indicator of initialization state, and (2) as a helper to encode shadow memory bit offsets
- This dual-purpose approach can serve as a starting point for exploring further methods to enhance other sanitizers.

Related Work

- **CombiSan**: CombiSan detects multiple classes of memory and undefined behavior errors simultaneously.
- **QMSan**: By leveraging QEMU's dynamic binary translation, QMSan efficiently detects Uninitialized Use of Memory (UUM) errors without requiring source-based recompilation.

Future Work

1. **KCSAN: Optimizing Detection.** Use memory tags as access locks to indicate if a memory location is being written by another thread.
2. **KUBSAN: Accelerating Checking.** Leverage tags to encode type information or bounds metadata, enabling faster verification of type confusion and array bounds violations.

Helsinki System Security Lab (HSSL)

HSSL drives renewal and mastery in the field of platform and device related security technologies, especially for Huawei consumer devices such as mobile phones, laptops, televisions and automotive. We do research in topics such as hardware-assisted isolation and integrity, as well as in operating system protection (hypervisor, TEE, secure enclaves and kernel hardening). We also carry expertise in cryptography and systems security functionality such as device key management (PKI), device attestation and key-store solutions.

HUAWEI