NanoTag: Systems Support for Efficient Byte-Granular Overflow Detection on ARM MTE

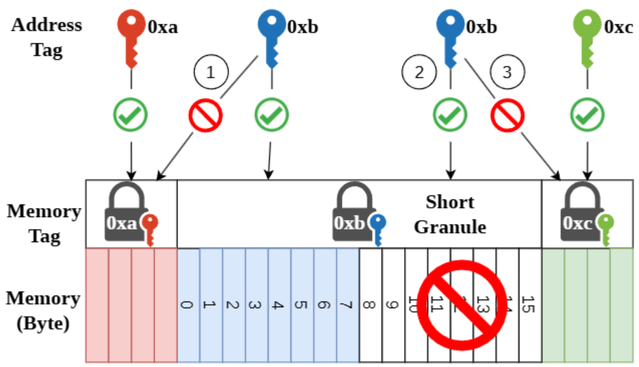
Mingkai Li, Hang Ye, Joseph Devietti, Suman Jana, Tanvir Ahmed Khan

Key Takeaway: Near-ASan bug detection (~97%) with MTE-like overhead (~12%) on real ARM MTE hardware.

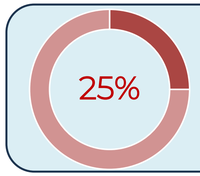
Motivation

- Memory safety bugs are the leading source of vulnerabilities (e.g., ~72% of 0-day exploits in 2025, according to Google's Project Zero).
- Companies like Google and Apple aim to ensure memory safety efficiently using ARM's Memory Tagging Extension (MTE).

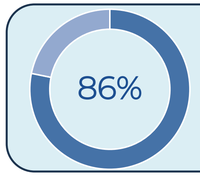
- MTE detects memory safety bugs across 16-byte tag granularity but ignores overflows inside a 16-byte granule.



ARM's Memory Tagging Extension (MTE) misses important bugs due to its 16-byte granularity:



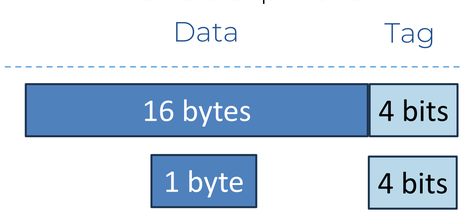
In NIST Juliet test suite, ARM MTE misses **25.40%** of heap-based buffer overflows, while ASan can detect almost all.



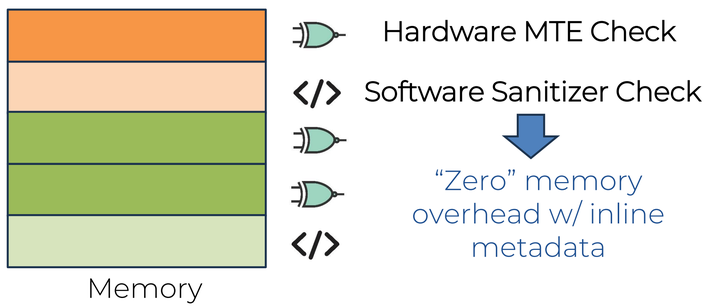
86.57% of memory allocations in SPEC CPU 2017 are potentially vulnerable to these overflows MTE misses.

Byte-Granular Detection

- Changing hardware to detect byte-granular overflows is expensive:



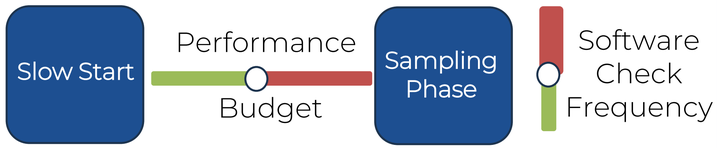
- Therefore, NANOTAG uses software checks when hardware checks are insufficient:



- However, software checks are significantly slower than hardware checks.

Explicit Detection-Performance Tradeoff

- NANOTAG introduces explicit detection-performance tradeoff via controllable knobs



Use case: in-house testing (e.g., fuzzing)
Interesting inputs: prioritize sanitizer accuracy
Repetitive inputs: prioritize fuzzing throughput

Evaluation Results

1) Bug Detection Capability*

	ASan	MTE (ASYNC)	MTE (SYNC)	NANOTAG
Heap-based Buffer Overflow	98.66%	75.60%	75.69%	97.57%

*: Evaluations based on NIST Juliet test suite.

2) Run-Time Overhead

	NANOTAG	ASan
SPEC CPU 2017 (geomean)	12.50%	95.11%
Real-world applications	Up to 12.35%	153.90%
MAGMA (fuzzing)	15.86%	111.20%
Geekbench 6 (closed-source)	NANOTAG 4.99%	Valgrind 1348.60%

Main Result

NANOTAG achieves **near-ASan** bug detection capability with **MTE-like** run-time overhead on real ARM MTE hardware.

Open-source and evaluated!

ARTIFACT EVALUATED IEEE S&P AVAILABLE

ARTIFACT EVALUATED IEEE S&P FUNCTIONAL

Mingkai Li, Columbia University
mailto: mingkai.li@columbia.edu
website: mingkai-li.github.io