

Porting LeakSanitizer:

A beginner's guide

Francis Ricci

Software Engineer, Facebook

Background

Heap Checker vs LeakSanitizer

Advantages of LeakSanitizer

- Performance
- Direct vs Indirect leak reports
- Thread-local data handling
- Suppressions

Heap Checker vs LeakSanitizer

Heap Checker supported platforms

	Linux	FreeBSD	Darwin	Android	Windows	Fuchsia
x86_64	Green	Yellow	Yellow	Green	Red	Red
x86	Green	Yellow	Yellow	Green	Red	White
aarch64	Green	Yellow	Yellow	Green	Red	Red
arm	Green	Yellow	Yellow	Green	Red	White
powerpc	Green	Yellow	Yellow	Green	Red	White
mips	Green	Yellow	White	Green	Red	White

Heap Checker vs LeakSanitizer

LeakSanitizer supported platforms

	Linux	FreeBSD	Darwin	Android	Windows	Fuchsia
x86_64	Green	Red	Green	Yellow	Red	Red
x86	Green	Red	Yellow	Yellow	Red	White
aarch64	Green	Red	Yellow	Yellow	Red	Red
arm	Green	Red	Red	Yellow	Red	White
powerpc	Green	Red	Red	Red	Red	White
mips	Green	Red	White	Red	Red	White

Heap Checker vs LeakSanitizer

AddressSanitizer supported platforms

	Linux	FreeBSD	Darwin	Android	Windows	Fuchsia
x86_64	Green	Green	Green	Green	Green	Green
x86	Green	Green	Green	Green	Green	White
aarch64	Green	Red	Green	Green	Red	Green
arm	Green	Red	Green	Green	Red	White
powerpc	Green	Red	Red	Red	Red	White
mips	Green	Green	White	Red	Red	White

Implementation

AddressSanitizer -> LeakSanitizer

Additional functionality required for leak checking

- Thread suspension
- Memory map
 - Thread-local storage
 - Static and global variables
- Platform-specific data sections

Thread suspension

Scan registers and stacks for heap pointers

- Suspend threads
 - Linux - `ptrace()`
 - Darwin - `thread_suspend()`
- Thread state parsing
 - Linux - `ptrace()`
 - Darwin - `thread_get_state()`

Memory map

Scan global data regions for pointers

- Generate map
 - Linux - `/proc/maps`
 - Darwin - `vm_region_recurse()/_dyld_get_image_header()`
- Data
 - Linux - `dl_phdr_info`
 - Darwin - `segment_command`
- Thread-local storage
 - `%fs/%gs`

Platform-specific data

Allocations requiring special handling

- Linux
 - Linker allocations (dynamic TLS blocks)
- Darwin
 - Kernel alloc once page
 - mmap'd regions

Testing

Testing LeakSanitizer

- compiler-rt
 - LSan test suite
 - ASan test suite with `DETECT_LEAKS=1`
- llvm+clang
 - `DETECT_LEAKS=1` for bootstrapped ASan builds
- Very large internal project
 - asm, c, c++, objective-c, swift
 - Compare behavior with LSan on Linux
 - Compare behavior with gperftools HeapChecker