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

<table>
<thead>
<tr>
<th>Platform</th>
<th>Linux</th>
<th>FreeBSD</th>
<th>Darwin</th>
<th>Android</th>
<th>Windows</th>
<th>Fuchsia</th>
</tr>
</thead>
<tbody>
<tr>
<td>x86_64</td>
<td>green</td>
<td>yellow</td>
<td>green</td>
<td>green</td>
<td>red</td>
<td>red</td>
</tr>
<tr>
<td>x86</td>
<td>green</td>
<td>green</td>
<td>green</td>
<td>green</td>
<td>red</td>
<td>green</td>
</tr>
<tr>
<td>aarch64</td>
<td>green</td>
<td>green</td>
<td>green</td>
<td>green</td>
<td>red</td>
<td>red</td>
</tr>
<tr>
<td>arm</td>
<td>green</td>
<td>green</td>
<td>green</td>
<td>green</td>
<td>red</td>
<td>green</td>
</tr>
<tr>
<td>powerpc</td>
<td>green</td>
<td>green</td>
<td>green</td>
<td>green</td>
<td>red</td>
<td>green</td>
</tr>
<tr>
<td>mips</td>
<td>green</td>
<td>green</td>
<td>white</td>
<td>green</td>
<td>red</td>
<td>green</td>
</tr>
</tbody>
</table>
## Heap Checker vs LeakSanitizer

LeakSanitizer supported platforms

<table>
<thead>
<tr>
<th>x86_64</th>
<th>Linux</th>
<th>FreeBSD</th>
<th>Darwin</th>
<th>Android</th>
<th>Windows</th>
<th>Fuchsia</th>
</tr>
</thead>
<tbody>
<tr>
<td>x86</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aarch64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>arm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>powerpc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mips</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Heap Checker vs LeakSanitizer

## AddressSanitizer supported platforms

<table>
<thead>
<tr>
<th></th>
<th>Linux</th>
<th>FreeBSD</th>
<th>Darwin</th>
<th>Android</th>
<th>Windows</th>
<th>Fuchsia</th>
</tr>
</thead>
<tbody>
<tr>
<td>x86_64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x86</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aarch64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>arm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>powerpc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mips</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
  - \texttt{asm, c, c++, objective-c, swift}
  - Compare behavior with LSan on Linux
  - Compare behavior with gperftools HeapChecker