A method to detect memory leaks & corruption

• Who are we?
• Why another tool?
• Tool features
• The idea
• Implementation
• Limitations
• Download
• Questions

References:
• Glibc Manual (3.2.2.9 Heap Consistency Checking)
• http://duma.sourceforge.net/
• http://valgrind.org/
• http://sourceware.org/binutils/docs-2.20/ld/index.html
• http://g.oswego.edu/dl/html/malloc.html

Ravi Sankar Guntur
ravisankar.g@gmail.com
27th Oct 2010
Who are we?

Part of team that works at Samsung India, Bangalore, for developing a **Smart-phone platform** based on GNU/Linux.

- **Platform**
  - ARM Processor.
  - 256 MB main memory.

- Major portion of debugging time spent on fixing **memory leaks and memory corruption** issues.

Vodafone - Samsung H1
Why we wanted another tool?

Tools we tried,
- Memory leaks
  - GNU libc’s mtrace
  - Valgrind – memcheck
  - memprof
- Memory corruptions
  - GNU libc’s MALLOC_CHECK_=2
  - DUMA (efence)
  - Valgrind - memcheck
- Issues
  - Huge memory overhead.
  - No support for GUI scenario based testing
  - Separate tools for memory leak and corruption
  - No support of call graph
Tool features

• Less memory overhead
• Provides call graph
• Support for scenario based memory leak testing
• Single tool to detect memory leaks and heap consistency

Memory leak report...

```
block of 18 bytes was not freed. (id: 0x12740)
/usr/lib/lsbafemem.so(malloc+0x70f0)[0x4002c174]
/lib/ld.so.6/+0x25954[0x4009fa54]
/lib/ld.so.6(bindtextdomain+0x11c)[0x4009fc9c]
/lib/ld.so.6(__libc_start_main+0x118)[0x4008f4c4]
```

Memory corruption report...

```
window:2238:Mon Oct 18 16:02:38 2010
memcpy-error: 0x3521c8 Allowed 8B, Needs 33B
```

backtrace

```
Backtrace...
/usr/lib/lsbafemem.so(0x7d898)[0x4002a898]
/usr/lib/lsbafemem.so(_bt+0x10)[0x4002a8f8]
/usr/lib/lsbafemem.so(memcpy+0xe8)[0x4002afb4]
/usr/lib/compiler/lib/segmentation_detection.c.usebacktrace+0x4f8[0x91941]
/usr/lib/libobject.so.0.0.0.g_closure_marshall_VOID_VOID+0x8c[0x422c1224]
/usr/lib/libobject.so.0.0.0.g_closure_marshall_VOID_VOID+0x10[0x422c9d34]
/usr/lib/libobject.so.0.0.0.g_closure_marshall_VOID_VOID+0x20[0x422c9d88]
/usr/bin/window(main+0x28)[0x9388]
/lib/ld.so.6(__libc_start_main+0x118)[0x4008f4c4]
```
The Idea – Memory Leak

“For every allocated block add Header and Footer. Add size and caller information in the Header”

“Erase the Header & Footer, before de-allocating the block”

”Scan the heap region for yet un-freed blocks and construct the call graph for every block found”

### The implementation

**Buffer typedef**

<table>
<thead>
<tr>
<th>Size</th>
<th>Header Sig1</th>
<th>Header Sig2</th>
<th>MODE</th>
<th># of Frames</th>
<th>Frame1</th>
<th>----</th>
<th>Frame 30</th>
<th>User Data</th>
<th>Footer Sig1</th>
<th>Footer Sig2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Buffer allocated when memory leak check is ON**

- Size: 28
- MODE: 1
- # of Frames: 12
- Frame1: 0x400 01234
- Frame 30: NULL
- User Data: 0xcafe babe
- Footer Sig1: 0xdeaf feed

---

**Buffer allocated when memory leak check is OFF**

- Size: 80
- MODE: 0
- # of Frames: 12
- Frame1: 0x400 01234
- Frame 30: NULL
- User Data: 0xcafe babe
- Footer Sig1: 0xdeaf feed

---

**Buffer allocated when memory leak check is ON**

**Buffer allocated when memory leak check is OFF**
The implementation – Memory leak

Generate memory leak report

Example heap layout

<table>
<thead>
<tr>
<th>Block1</th>
<th>Block2</th>
<th>Block3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free</td>
<td>Block5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Block5</td>
<td></td>
</tr>
<tr>
<td>Block6</td>
<td>Free</td>
<td></td>
</tr>
</tbody>
</table>

Flowchart:
- Generate Report
  - Current heap = Heap Base
  - Current heap <= Heap End
    - Yes: Get next allocated block
    - No: MODE == 1
      - Yes: Get Symbols from Frame Ptrs
      - No: Write leak report to log
  - Current Heap = End of current block
  - Stop
The Idea – Memory corruption

“One of the source of memory corruption is wrong usage of parameters to lib C’s string manipulation functions”

“use LD_PRELOAD to preload DSO of modified functions”

“Given destination buffer, get the size from Header and check for possible memory corruption”
To check heap consistency,
• Preloaded string wrappers check if the destination address is from heap region or not.
  • If from heap
    • Checks the validity of the buffer.
    • Checks if number of bytes > allocated size.
      • If yes, error details will be written to log file and SIGSEG will be raised.
      • If no, proceed normally
  • if not from heap, proceed normally
• realloc, calloc, and free will check header and footer for integrity.
Limitations

- Shell script sets up the environment variables like LD_PRELOAD, LEAK_MODE, G_SLICE and launches the debugged program.
- “-fno-omit-frame-pointer” is needed for backtrace()
- if no “-rdynamic”, use addr2line to convert VMA to Symbol name.
Download

- Integrated tool with couple of bug fixes is not yet uploaded to public domain. (contact the author to check the latest status)

- Separate tools to detect memory corruption and leaks are available at,
  - `git clone git://git.savannah.nongnu.org/safeheap.git`
  - `git clone git://git.savannah.nongnu.org/memleak.git`
Questions
END...