Gold Standard Benchmark for Static Source Code Analyzers

Kestrel Technology, LLC
Dr. Henny B. Sipma

Sep 16, 2013
Kestrel Technology

**Founded:** 2000

**Location:** Palo Alto, CA

**Core activity:** Sound static analysis (mathematical proof of safety)

**Tool:** CodeHawk

**Languages supported:** C, Java, x86 executables

**Underlying technology:** Abstract interpretation (Cousot, Cousot, 1977)

**Properties**

- **C:** Language-level properties: memory safety, null-dereference
  (application-independent, mathematically well-defined properties)

- **Java:** Information flow analysis, resource analysis, integer overflow, error handling

- **X86:** Memory safety, information extraction
Cement is everywhere. Our very lives depend on cement, yet we don't worry about it: such is the legacy of Joseph Bazalgette, who introduced measurement and strict quality controls.

Like cement, software is everywhere: our very lives are becoming more and more dependent on software.

Unlike cement, for more than 50 years software has been continuously released and added to the national infrastructure, plagued by design and implementation defects that were largely detectable and preventable, but were not. Why not?

(from Geekonomics, David Rice)

Physically safe in the largest buildings, but ... not safe from continuous attacks from all over the world.
Customer Need

We need the ability to measure software quality

We need the ability to measure the effectiveness of the tools that are used for software assurance

in terms of

False positives: bugs that turn out not to be bugs

AND

False negatives: bugs that are missed
Problems with measurement

From the 2012 Coverity Scan Report (May 2013):

<table>
<thead>
<tr>
<th>Coverity Scan Report Year</th>
<th>Average Defect Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008</td>
<td>.30</td>
</tr>
<tr>
<td>2009</td>
<td>.25</td>
</tr>
<tr>
<td>2010</td>
<td>.81</td>
</tr>
<tr>
<td>2011</td>
<td>.45</td>
</tr>
<tr>
<td>2012</td>
<td>.69</td>
</tr>
</tbody>
</table>

(indicator of defects per 1000 LOC)

Did software get worse?
Did detection technology get better?
Some combination of the two?

What is the "real" number?
What is a defect?
Compounding problem: Economic incentives favor less assurance

Reality

Customer sees

Tool A    Tool B

false positives

false negatives

.... and will choose Tool A
Our Solution:
Measure memory-safety defects

Define Defect

mathematical **definition** based on
- C standard semantics (37 cases of memory-related undefined behavior)
- Semantic explication of C semantics by George Necula (Berkeley, 2002)
  - covers all of C plus gcc extensions

Measure Defects

mathematical **proof** based on
- Primary proof obligations (for all memory accesses and related instructions)
- Sound static analysis to discharge proof obligations of safe memory accesses
- Demonstrate vulnerability in remaining cases
Our Solution: Measure memory-safety defects

Define Defect

mathematical definition based on
- C standard semantics (37 cases of memory-related undefined behavior)
- Semantic explication of C semantics by George Necula (Berkeley, 2002)
  - covers all of C plus gcc extensions

Measure Defects

mathematical proof based on
- Primary proof obligations (for all memory accesses and related instructions)
- Sound static analysis to discharge proof obligations of safe memory accesses
- Demonstrate vulnerability in remaining cases

DIFFICULT
Kestrel Technology's Core Capability:

powerful and scalable, sound static analysis engine:

- based on theory of abstract interpretation (Cousot, Cousot 1977)
- generates invariants (over-approximation of program behaviors for all possible inputs)

Technology: Distributed Proof Structure

- structural induction over the control flow graph of all functions
- assume-guarantee reasoning based on
  - function summaries (contracts)
  - data structure invariants
  - global data invariants
- hierarchy of primary and secondary proof obligations
- local invariant generation
- whole-program pointer analysis
## Apply to six benchmark programs

<table>
<thead>
<tr>
<th>Program</th>
<th>Description</th>
<th>Files</th>
<th>Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>lighttpd</strong></td>
<td>Webserver designed and optimized for high-performance environments used by several popular websites, including YouTube and Wikipedia</td>
<td>144</td>
<td>38 KLOC</td>
</tr>
<tr>
<td><strong>nagios</strong></td>
<td>Tool for IT-infrastructure monitoring with more than a million users, including Comcast, Philips, Siemens, Thales, McAfee</td>
<td>73</td>
<td>65 KLOC</td>
</tr>
<tr>
<td><strong>naim</strong></td>
<td>Console chat client for AOL instant messaging (AIM) and internet relay chat (IRC)</td>
<td>44</td>
<td>23 KLOC</td>
</tr>
<tr>
<td><strong>irssi</strong></td>
<td>Terminal-based IRC client for UNIX systems</td>
<td>347</td>
<td>53 KLOC</td>
</tr>
<tr>
<td><strong>pvm</strong></td>
<td>Package that lets heterogeneous collections of Unix and Windows computers to be used as a single large parallel computer, developed by Oak Ridge Nat'l Labs</td>
<td>320</td>
<td>72 KLOC</td>
</tr>
<tr>
<td><strong>dovecot</strong></td>
<td>Open-source IMAP and POP3 email server for Linux (written with security in mind)</td>
<td>1,111</td>
<td>119 KLOC</td>
</tr>
<tr>
<td>Program</td>
<td>Description</td>
<td>Files</td>
<td>KLOC</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>lighttpd</td>
<td>Webserver designed and optimized for high-performance environments used by several popular websites, including YouTube and Wikipedia</td>
<td>144</td>
<td>38</td>
</tr>
<tr>
<td>nagios</td>
<td>Tool for IT-infrastructure monitoring with more than a million users, including Comcast, Philips, Siemens, Thales, McAfee</td>
<td>73</td>
<td>65</td>
</tr>
<tr>
<td>naim</td>
<td>Console chat client for AOL instant messaging (AIM) and internet relay chat (IRC)</td>
<td>44</td>
<td>23</td>
</tr>
<tr>
<td>irssi</td>
<td>Terminal-based IRC client for UNIX systems</td>
<td>347</td>
<td>53</td>
</tr>
<tr>
<td>pvm</td>
<td>Package that lets heterogeneous collections of Unix and Windows computers to be used as a single large parallel computer, developed by Oak Ridge Nat'l Labs</td>
<td>320</td>
<td>72</td>
</tr>
<tr>
<td>dovecot</td>
<td>Open-source IMAP and POP3 email server for Linux (written with security in mind)</td>
<td>1,111</td>
<td>119</td>
</tr>
</tbody>
</table>
Approach: Architecture

- **Interactive Proof Assistant**
- **CHCIL parser**
- **XML representation of code and proofs**
- **Scoring Tool**
- **Progress Monitor**
- **3rd party tool results**
- **Tool Report**

**Diagram Explained:**
- The C program goes through a CHCIL parser which creates compilation units and linking units.
- Compilation units and linking units lead to the XML representation of code and proofs.
- The XML library function summaries are used to further process the XML representation.
- Interactive Proof Assistant and Progress Monitor interact with the Scoring Tool, which generates a Tool Report.
- The tool results are integrated into the Scoring Tool to enhance its functionality.
Approach: Scoring tool

**Scoring Tool:**
- Import results for all primary and secondary proof obligations
- Import 3rd party analysis tool results
- Compare
- Establish correspondence and report
  - false negatives
  - false positives
  - adequacy of bug placement
- Identify strengths and weaknesses of 3rd party analysis tool

**Programs:**
- lighttpd
- nagios
- naim
- irssi
- pvm
- dovecot

Apply to results from any static analysis tool in the SWAMP
Benefit: Ability to grade tools
Collaboration (rather than Competition)

Other government organizations:
- NIST (Dr. Paul E. Black)

Vendors of bug-finding tools:
- Offer scoring tool as a service to
  - Identify strengths and weaknesses in bug-finding tools
  - Improve bug-finding tools
Current Status: Accomplishments

- Theoretical foundations of memory-safety defect measurement
  - shared and discussed with Dr. Paul E. Black (NIST)
  - relationship to 27 CWE's
  - illustrative application to lighttpd

- Software architecture and implementation (in progress)
  - xml representation of CIL semantic constructs
  - xml library function summaries
  - interactive proof assistant
  - progress monitor

- Benchmark programs (in progress)
  - xml representation of lighttpd
Schedule and Milestones

XML representation of code and proofs

IPA
PM

lighttpd
nagios
irssi
pvm
dovecot

Sep
Dec
Mar
June
Sep

Scoring Tool

Tool Report

9/12/2013
2013 DHS S&T/DoD ASD (R&E) CYBER SECURITY SBIR WORKSHOP
SWAMP

Ability to score and evaluate static analysis tools

Preprocessed programs to
- lower entry barrier for new tool developers
- facilitate collaborative analysis
Contact Information

Team:
• Henny Sipma, Principal Investigator
• Rich Barry, Program Manager
• Eric Bush
• Arnaud Venet (Consultant)

Contact Information: sipma@kestreltechnology.com

Kestrel Technology, LLC
3260 Hillview Ave
Palo Alto, CA 94304
tel. (650) 320 8487