Lecture 3. Implementation and Experimentation

Wei Le

2014.9
Experiments

Theories can be refuted by experiments. This distinguishes science from religion.

“A theory which cannot be mortally endangered cannot be alive”

W. A. H. Rushton

Experiments can be reproduced by others, in order to verify the results.
Research Method - Hypothesis Testing

- Problem
- Hypothesis:
  - Experience, observation, insights, prediction, educated guess, never known before, e.g., *demand-driven analysis* improves the scalability of buffer overflow detection
  - A model that tries to explain observations and enable predictions, and can be tested experimentally
- Null-hypothesis: e.g., *demand-driven analysis* cannot improve the scalability of buffer overflow detection
- Reject null-hypothesis (using a counter example) to support hypothesis: NOTE, you really cannot prove hypothesis always is true
- *Controlled Experiments*
Experimental Evaluation - Steps

- Define Goal(s)
- Implementation: Prototyping the NEW algorithm
- Collect data
- Sample and inspecting results to make sure the confidence, correct the bugs if needed
- Data presentation and summary (statistics): tables, figures, diagrams, R, Python to help understand and draw conclusions
- Data analysis:
  - how data support your goal and hypothesis?
  - What are the corner cases? Why they are interesting?
  - Gain further insights
Experimental Design

- Reproducibility: understand/identify exact conditions that enable reproduce the results
- Threats to validity (if you are establishing a causal relationship):
  - Internal validity: Inferences are said to possess internal validity if a causal relation between two variables is properly demonstrated
  - External validity: A threat to external validity is an explanation of how you might be wrong in making a generalization. Generally, generalizability is limited when the cause (i.e. the independent variable) depends on other factors
  - Construct validity: whether what you did for the program was what you wanted to do or whether what you observed was what you wanted to observe
Selection of Benchmarks

- Bug detection capabilities: programs with bugs, programs with seeded bugs, programs with mutants
- Performance and scalability: large programs, a variety of programs, programs with many users
- Test input capabilities: programs with given test input
- Understanding and analyzing software changes and versions: multiple versions of software, patches, software productline
- Analyzing special software: Android applications
- A study of ICSE/FSE papers: How people select benchmarks?
Benchmarks

- Buffer Overflow Benchmarks [4]
- Bugbench [3]
- SIR Benchmarks
- BegBunch: benchmarks for C bug detection tools [1]
- Spec: http://www.spec.org/
- Benchmarks for evaluating automatic program repairing techniques
## Benchmarks

**Bugbench**

<table>
<thead>
<tr>
<th>Name</th>
<th>Program</th>
<th>Source</th>
<th>Description</th>
<th>Line of Code</th>
<th>Bug Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCOM</td>
<td>ncompress-4.2.4</td>
<td>Red Hat Linux</td>
<td>file (de)compression</td>
<td>1.9K</td>
<td>Stack Smash</td>
</tr>
<tr>
<td>POLY</td>
<td>polymorph-0.4.0</td>
<td>GNU</td>
<td>file system &quot;unixier&quot; (Win32 to Unix filename converter)</td>
<td>0.7K</td>
<td>Stack Smash &amp; Global Buffer Overflow</td>
</tr>
<tr>
<td>GZIP</td>
<td>gzip-1.2.4</td>
<td>GNU</td>
<td>file (de)compression</td>
<td>8.2K</td>
<td>Global Buffer Overflow</td>
</tr>
<tr>
<td>MAN</td>
<td>man-1.5h1</td>
<td>Red Hat Linux</td>
<td>documentation tools</td>
<td>4.7K</td>
<td>Global Buffer Overflow</td>
</tr>
<tr>
<td>GO</td>
<td>099.go</td>
<td>SPEC95</td>
<td>game playing (Artificial Intelligent)</td>
<td>29.6K</td>
<td>Global Buffer Overflow</td>
</tr>
<tr>
<td>COMP</td>
<td>129.compress</td>
<td>SPEC95</td>
<td>file compression</td>
<td>2.0K</td>
<td>Global Buffer Overflow</td>
</tr>
<tr>
<td>BC</td>
<td>bc-1.06</td>
<td>GNU</td>
<td>interactive algebraic language</td>
<td>17.0K</td>
<td>Heap Buffer Overflow</td>
</tr>
<tr>
<td>SQUID</td>
<td>squid-2.3</td>
<td>Squid</td>
<td>web proxy cache server</td>
<td>93.5K</td>
<td>Heap Buffer Overflow</td>
</tr>
<tr>
<td>CALB</td>
<td>cachelib</td>
<td>UIUC</td>
<td>cache management library</td>
<td>6.6K</td>
<td>Uninitialized Read</td>
</tr>
<tr>
<td>CVS</td>
<td>cvs-1.11.4</td>
<td>GNU</td>
<td>version control</td>
<td>114.5K</td>
<td>Double Free</td>
</tr>
<tr>
<td>YPSV</td>
<td>ypserv-2.2</td>
<td>Linux NIS</td>
<td>NIS server</td>
<td>11.4K</td>
<td>Memory Leak</td>
</tr>
<tr>
<td>PFTP</td>
<td>proftpd-1.2.9</td>
<td>ProFTPD</td>
<td>ftp server</td>
<td>68.9K</td>
<td>Memory Leak</td>
</tr>
<tr>
<td>SQUID2</td>
<td>squid-2.4</td>
<td>Squid</td>
<td>web proxy cache</td>
<td>104.6K</td>
<td>Memory Leak</td>
</tr>
<tr>
<td>HTPD1</td>
<td>httpd-2.0.49</td>
<td>Apache</td>
<td>HTTP server</td>
<td>224K</td>
<td>Data Race</td>
</tr>
<tr>
<td>MYSQL1</td>
<td>mysql-4.1.1</td>
<td>MySQL</td>
<td>database</td>
<td>1028K</td>
<td>Data Race</td>
</tr>
<tr>
<td>MYSQL2</td>
<td>mysql-3.23.56</td>
<td>MySQL</td>
<td>database</td>
<td>514K</td>
<td>Atomicity</td>
</tr>
<tr>
<td>MYSQL3</td>
<td>mysql-4.1.1</td>
<td>MySQL</td>
<td>database</td>
<td>1028K</td>
<td>Atomicity</td>
</tr>
<tr>
<td>PSQL</td>
<td>postgresql-7.4.2</td>
<td>PostgreSQL</td>
<td>database</td>
<td>559K</td>
<td>Semantic Bug</td>
</tr>
<tr>
<td>HTPD2</td>
<td>httpd2.0.49</td>
<td>Apache</td>
<td>HTTP server</td>
<td>224K</td>
<td>Semantic Bug</td>
</tr>
</tbody>
</table>
## Benchmarks

### BegBunch

<table>
<thead>
<tr>
<th>Size</th>
<th>Benchmark</th>
<th># Programs</th>
<th># Bugs</th>
<th>LOC min</th>
<th>LOC max</th>
<th>LOC avg</th>
<th>Harness</th>
<th>Language</th>
<th>Multiplatform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>Zitser</td>
<td>14</td>
<td>83</td>
<td>175</td>
<td>1.5K</td>
<td>657</td>
<td>No</td>
<td>C</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Kratkiewicz</td>
<td>291 x 4</td>
<td>873</td>
<td>6</td>
<td>27</td>
<td>14</td>
<td>No</td>
<td>C</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>SAMATE</td>
<td>375</td>
<td>408</td>
<td>20</td>
<td>1.1K</td>
<td>90</td>
<td>No</td>
<td>C,C++,Java,PHP</td>
<td>Yes</td>
</tr>
<tr>
<td>Large</td>
<td>BugBench</td>
<td>10</td>
<td>19</td>
<td>735</td>
<td>692K</td>
<td>149K</td>
<td>No</td>
<td>C</td>
<td>Linux</td>
</tr>
<tr>
<td></td>
<td>Faultbench v0.1</td>
<td>6</td>
<td>11</td>
<td>1,276</td>
<td>25K</td>
<td>7K</td>
<td>No</td>
<td>Java</td>
<td>Yes</td>
</tr>
</tbody>
</table>
How to Implement a Program Analysis Tool

Compiler passes and phases: Program analysis tools and compilers
(figure on board)
Platforms and Tools

How to Implement a Program Analysis Tool

- Phoenix, LLVM, Soot
- Tools: Klee, Crest – Concolic testing, Saturn – bug detection, MOPS – verify security property, Clang
Cristina Cifuentes, Christian Hoermann, Nathan Keynes, Lian Li, Simon Long, Erica Mealy, Michael Mounteney, and Bernhard Scholz.

Begbunch: Benchmarking for C bug detection tools.

Nicholas Jalbert, Cristiano Pereira, Gilles Pokam, and Koushik Sen.
Radbench: A concurrency bug benchmark suite.

Shan Lu, Zhenmin Li, Feng Qin, Lin Tan, Pin Zhou, and Yuanyuan Zhou.
Bugbench: Benchmarks for evaluating bug detection tools.

Testing static analysis tools using exploitable buffer overflows from open source code.