Have Things Changed Now?

- An Empirical Study of Bug Characteristics in Modern Open Source Software

Zhenmin Li, Lin Tan, Xuanhui Wang, Shan Lu, Yuanyuan Zhou and Chengxiang Zhai

University of Illinois, Urbana-Champaign
Bugs account for 40% of system failures.

Designing debugging tools requires a good understanding of bug characteristics.

Many empirical studies have been performed.

Why another empirical study?
Things have changed

Many empirical studies have been performed over 10 years ago
Things have changed

- Many empirical studies have been performed over 10 years ago
- More effective modern debugging tools - Valgrind, Purify, Coverity, etc
Things have changed

- Many empirical studies have been performed over 10 years ago
- More effective modern debugging tools - Valgrind, Purify, Coverity, etc
- Fewer memory bugs in release code?
Things have changed

- Many empirical studies have been performed over 10 years ago
- More effective modern debugging tools - Valgrind, Purify, Coverity, etc
  - Fewer memory bugs in release code?
- Rising security concerns
Things have changed

- Many empirical studies have been performed over 10 years ago
- More effective modern debugging tools - Valgrind, Purify, Coverity, etc
- Fewer memory bugs in release code?
- Rising security concerns
  - More security-related bugs?
Things have changed

- Emphasis on user friendly interfaces
Things have changed

- Emphasis on user friendly interfaces
  - Have GUI bugs become more pervasive?
Things have changed

- Emphasis on user friendly interfaces
  - Have GUI bugs become more pervasive?
- Software architecture shift - Multithreaded/Multiprocessed
Things have changed

- Emphasis on user friendly interfaces
  - Have GUI bugs become more pervasive?
- Software architecture shift - Multithreaded/Multiprocessed
  - More concurrency bugs?
Things have changed

- Emphasis on user friendly interfaces
  - Have GUI bugs become more pervasive?
- Software architecture shift - Multithreaded/Multiprocessed
  - More concurrency bugs?
- Need a new empirical study to answer these questions
Our Contribution

- Studied two large and popular OSS projects

<table>
<thead>
<tr>
<th></th>
<th>Mozilla</th>
<th>Apache</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Our Contribution

- Studied two large and popular OSS projects

<table>
<thead>
<tr>
<th>Project Type</th>
<th>Mozilla</th>
<th>Apache</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Client</td>
<td>Server</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mozilla | Apache
---|---
Client | Server
Our Contribution

- Studied two large and popular OSS projects

<table>
<thead>
<tr>
<th></th>
<th>Mozilla</th>
<th>Apache</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project Type</strong></td>
<td>Client</td>
<td>Server</td>
</tr>
<tr>
<td><strong>Major Language</strong></td>
<td>C &amp; C++</td>
<td>C</td>
</tr>
</tbody>
</table>
Our Contribution

- Studied two large and popular OSS projects

<table>
<thead>
<tr>
<th></th>
<th>Mozilla</th>
<th>Apache</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Type</td>
<td>Client</td>
<td>Server</td>
</tr>
<tr>
<td>Major Language</td>
<td>C &amp; C++</td>
<td>C</td>
</tr>
<tr>
<td>Code Size (MLOC)</td>
<td>4</td>
<td>0.3</td>
</tr>
</tbody>
</table>
Our Contribution

- Studied two large and popular OSS projects

<table>
<thead>
<tr>
<th></th>
<th>Mozilla</th>
<th>Apache</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project Type</strong></td>
<td>Client</td>
<td>Server</td>
</tr>
<tr>
<td><strong>Major Language</strong></td>
<td>C &amp; C++</td>
<td>C</td>
</tr>
<tr>
<td><strong>Code Size (MLOC)</strong></td>
<td>4</td>
<td>0.3</td>
</tr>
<tr>
<td><strong>No. of Releases</strong></td>
<td>~90</td>
<td>~90</td>
</tr>
</tbody>
</table>
Our Contribution

- Studied two large and popular OSS projects

<table>
<thead>
<tr>
<th>Project Type</th>
<th>Mozilla</th>
<th>Apache</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major Language</td>
<td>C &amp; C++</td>
<td>C</td>
</tr>
<tr>
<td>Code Size (MLOC)</td>
<td>4</td>
<td>0.3</td>
</tr>
<tr>
<td>No. of Releases</td>
<td>~90</td>
<td>~90</td>
</tr>
<tr>
<td>Bug DB start time</td>
<td>1998</td>
<td>2001</td>
</tr>
</tbody>
</table>

- Mozilla
- Apache

Mozilla

Project Type: Client
Major Language: C & C++
Code Size (MLOC): 4
No. of Releases: ~90
Bug DB start time: 1998

Apache

Project Type: Server
Major Language: C
Code Size (MLOC): 0.3
No. of Releases: ~90
Bug DB start time: 2001
Our Contribution

- Studied two large and popular OSS projects

<table>
<thead>
<tr>
<th></th>
<th>Mozilla</th>
<th>Apache</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Type</td>
<td>Client</td>
<td>Server</td>
</tr>
<tr>
<td>Major Language</td>
<td>C &amp; C++</td>
<td>C</td>
</tr>
<tr>
<td>Code Size (MLOC)</td>
<td>4</td>
<td>0.3</td>
</tr>
<tr>
<td>No. of Releases</td>
<td>~90</td>
<td>~90</td>
</tr>
<tr>
<td>Bug DB start time</td>
<td>1998</td>
<td>2001</td>
</tr>
</tbody>
</table>

- Manually collected and classified 709 bugs
Our Contribution

- Studied two large and popular OSS projects

<table>
<thead>
<tr>
<th></th>
<th>Mozilla</th>
<th>Apache</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Type</td>
<td>Client</td>
<td>Server</td>
</tr>
<tr>
<td>Major Language</td>
<td>C &amp; C++</td>
<td>C</td>
</tr>
<tr>
<td>Code Size (MLOC)</td>
<td>4</td>
<td>0.3</td>
</tr>
<tr>
<td>No. of Releases</td>
<td>~90</td>
<td>~90</td>
</tr>
<tr>
<td>Bug DB start time</td>
<td>1998</td>
<td>2001</td>
</tr>
</tbody>
</table>

- Manually collected and classified 709 bugs
- Automatically classified around 29,000 bugs
<table>
<thead>
<tr>
<th></th>
<th>Mozilla</th>
<th>Apache</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Our Bug Sources

<table>
<thead>
<tr>
<th>General</th>
<th>Mozilla</th>
<th>Apache</th>
</tr>
</thead>
<tbody>
<tr>
<td>Randomly sampled</td>
<td>548</td>
<td>209</td>
</tr>
</tbody>
</table>
# Our Bug Sources

<table>
<thead>
<tr>
<th>Source</th>
<th>Mozilla</th>
<th>Apache</th>
</tr>
</thead>
<tbody>
<tr>
<td>General - Randomly sampled</td>
<td>548</td>
<td>209</td>
</tr>
<tr>
<td>After filtering</td>
<td>264</td>
<td>98</td>
</tr>
</tbody>
</table>

After filtering: only fixed run-time bugs with known root causes
Our Bug Sources

<table>
<thead>
<tr>
<th>Source</th>
<th>Mozilla</th>
<th>Apache</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>548</td>
<td>209</td>
</tr>
<tr>
<td></td>
<td>Randomly sampled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>264</td>
<td>98</td>
</tr>
<tr>
<td>Security</td>
<td>193</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>All</td>
<td></td>
</tr>
</tbody>
</table>

After filtering: only fixed run-time bugs with known root causes
## Our Bug Sources

<table>
<thead>
<tr>
<th></th>
<th>Mozilla</th>
<th>Apache</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Randomly sampled</td>
<td>548</td>
<td>209</td>
</tr>
<tr>
<td>After filtering</td>
<td>264</td>
<td>98</td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>193</td>
<td>64</td>
</tr>
</tbody>
</table>

We use information retrieval techniques to collect concurrency bugs and manually classify 90 of them.

**After filtering: only fixed run-time bugs with known root causes**
## Our Bug Sources

<table>
<thead>
<tr>
<th></th>
<th>Mozilla</th>
<th>Apache</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Randomly sampled</td>
<td>548</td>
<td>209</td>
</tr>
<tr>
<td>After filtering</td>
<td>264</td>
<td>98</td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>193</td>
<td>64</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>709</td>
<td></td>
</tr>
</tbody>
</table>

We use information retrieval techniques to collect concurrency bugs and manually classify 90 of them.

**After filtering:** only fixed run-time bugs with known root causes.
Classification Dimensions

- Component
- Impact
- Root Cause
Every bug is classified in three dimensions.
Classification Dimensions

Every bug is classified in three dimensions.

- Component
- Impact
- Root Cause
  - Memory
  - Concurrency
  - Semantic
Classification Dimensions

Every bug is classified in three dimensions.
Every bug is classified in three dimensions.
Major Findings

1992 [Sullivan92]
Major Findings

- Memory bugs are decreasing.


- Memory bugs are decreasing.
Major Findings

- Memory bugs are decreasing.
- Semantic bugs are becoming increasingly dominant.

1992 [Sullivan92]  

2005
Major Findings

- Memory bugs are decreasing.
- Semantic bugs are becoming increasingly dominant.
- A few concurrency bugs probably because of underreporting.

Impact

Results are similar for both Mozilla and Apache

- Memory
- Concurrency
- Semantic
Impact

- 57.1% of crashes are caused by memory bugs.

Results are similar for both Mozilla and Apache

57.1% of crashes are caused by memory bugs.
57.1% of crashes are caused by memory bugs.
57.1% of crashes are caused by memory bugs.

Incorrect functionality dominates.

Results are similar for both Mozilla and Apache.

Impact
57.1% of crashes are caused by memory bugs.

Incorrect functionality dominates.
Impact

- 57.1% of crashes are caused by memory bugs.
- Incorrect functionality dominates.

Results are similar for both Mozilla and Apache.

See our paper for numbers with error range with 95% confidence level.
Component

0% 20% 40% 60% 80%

Core GUI Net I/O Others
Mozilla

Core GUI Net I/O Others
Apache
Component

- Core: 40.5%
- GUI: 52.7%
- Net Mozilla: 1.5%
- I/O: 2.3%
- Others: 3.0%
- Core: 76.5%
- GUI: 2.0%
- Net Apache: 8.2%
- I/O: 12.2%
- Others: 1.0%
Component

Client and server software have different bug characteristics.
Security Bugs - Trend

Number of bugs

- Mozilla
- Apache
Security Bugs - Trend

- Mozilla
- Apache

Only contain half a year's data

Number of bugs
Security Bugs - Trend

Number of bugs

Normalized Percentage

Only contain half a year’s data
Security Bugs - Trend

- Only contain half a year’s data

Number of bugs

Normalized Percentage
Security Bugs - Trend

Number of bugs

Normalized Percentage

Only contain half a year’s data
Security Bugs - Trend

- Security bugs are increasing.

Only contain half a year’s data

Security bugs are increasing.
Results are similar for both Mozilla and Apache.
AGAINST the belief that buffer overflows are the most common form of security vulnerabilities.

Results are similar for both Mozilla and Apache.
Automatic Classification

- Do the previous distributions hold for all bugs?
- Do we see the same trend across software evolution?
Automatic Classification

- Do the previous distributions hold for all bugs?
- Do we see the same trend across software evolution?
- Require more bugs or all of the reported bugs, which could be around 29,000
Automatically Classification

- Do the previous distributions hold for all bugs?
- Do we see the same trend across software evolution?
- Require more bugs or all of the reported bugs, which could be around 29,000
- Can NOT be done MANUALLY
Automatic Classification

- Do the previous distributions hold for all bugs?
- Do we see the same trend across software evolution?
- Require more bugs or all of the reported bugs, which could be around 29,000
- Can NOT be done MANUALLY
- Use machine learning techniques to automatically classify all 29,000 or so fixed run-time bugs in Mozilla (by June 2005)
Auto Classification Results

Automatic classification confirmed the distribution results of root causes and impacts (see our paper).
Auto Classification Results

Automatic classification confirmed the distribution results of root causes and impacts (see our paper).

![Graph showing distribution of Semantic and Memory]
Auto Classification Results

Automatic classification confirmed the distribution results of root causes and impacts (see our paper).

![Graph showing percentage distribution over years for Semantic and Memory categories.](image)
Auto Classification Results

Automatic classification confirmed the distribution results of root causes and impacts (see our paper).

Semantic bugs increase over time
Auto Classification Results

Automatic classification confirmed the distribution results of root causes and impacts (see our paper).

Semantic bugs increase over time
Auto Classification Results

Automatic classification confirmed the distribution results of root causes and impacts (see our paper).

- Semantic bugs increase over time
- Memory bugs decrease over time
Conclusions

- Manually studied 709 bugs and automatically classified 29,000 bugs
Conclusions

- Manually studied 709 bugs and automatically classified 29,000 bugs
- Memory bugs in release code decreased.
- Detection tools have a positive impact.
Conclusions

- Manually studied 709 bugs and automatically classified 29,000 bugs
- Memory bugs in release code decreased.
  - Detection tools have a positive impact.
- Semantic bugs are a major source of bugs, accounting for 81.1-86.7% of all classified bugs.
  - More effort should be applied to detecting and fixing semantic bugs.
Conclusions

- Most security vulnerabilities are caused by semantic bugs
Conclusions

- Most security vulnerabilities are caused by semantic bugs
- Client and server have different bug characteristics.
  - Need different testing support
Conclusions

- Most security vulnerabilities are caused by semantic bugs.
- Client and server have different bug characteristics.
  - Need different testing support.
- Concurrency bugs are hard to reproduce.
  - Need tools, e.g. FDR and BugNet, to help replay.
Conclusions

- Most security vulnerabilities are caused by semantic bugs.
- Client and server have different bug characteristics.
  - Need different testing support.
- Concurrency bugs are hard to reproduce.
  - Need tools, e.g. FDR and BugNet, to help replay.
- Correlation results and minor findings are shown in our paper.
Questions?
Thank you!