Razor: A Framework for Post-deployment Software Debloating

Chenxiong Qian, Hong Hu, Mansour Alharthi, Pak Ho (Simon) Chung, Taesoo Kim, Wenke Lee
Software Is Getting Bigger

Lines of Code vs. Linux Kernel Version

- 5M Lines of Code
- 17.5M Lines of Code
Software Is Bloated

➢ Software contains dead code.

Quach et al. (FEAST’17)
Software Is Bloated

➢ Software contains code that is never used by users.

![Dynamic Code Coverage](image)

Avg: 20.96%

Quach et al. (FEAST'17)
Bloated Code Increases Attack Surface

➢ Example1: HeartBleed
  ○ TLS heartbeat extension.
  ○ Not used by most users.
  ○ Enabled in default.

➢ Example2: CVE-2014-0038
  ○ `compat_sys_recvmmsg` handles `recvmsg` system call for x32 ABI.
  ○ x32 ABI takes advantage of the 64-bit environment while using 32-bit pointers for less overhead.
  ○ No such programs exist in real world!
  ○ X32 is enabled by default in all major distributions like Ubuntu!
Software Debloating

➢ All existing software debloating systems have the following limitations:

  ○ Require source code.
    ■ Source code is not always accessible to users.
    ■ It’s challenging and time-consuming to recompile source code.

  ○ Assume test cases are complete.
    ■ This assumption mostly fails in real world.
    ■ Impossible to provide complete test cases for a particular functionality.
Razor

➢ Performs code reduction for deployed **binaries**.

➢ Uses **heuristics** to infer related code for given test cases.
Overview

Razor

bloated binary -> Tracer (Dynamorio, Intel PIN, Intel PT) -> execution traces (decode) -> Path Finder (CFG) -> Generator (assembler, instrumenter, fault handler) -> debloated binary

test cases
Tracer

➢ Multiple tracers
  ○ Software-based tracers (Dynamorio, Intel PIN)
    ■ Complete trace
    ■ Significant overhead
  ○ Hardware-based tracer (Intel PT)
    ■ Small overhead
    ■ Incomplete trace
  ○ Programs under different tracing environments show divergent paths.

➢ The collected trace contains three parts:

Executed Blocks
[0x4005c0, 0x4005f2]
[0x400596,0x4005ae]
...

Conditional Branches
[0x4004e3: true]
[0x4004ee: false]
[0x400614: true, false]
...

Indirect Calls/Jumps
[0x400677, 0x4005e6#18, 0x4005f6#6
...
Path Finder

➢ Four Heuristics

- **zCode (zero code)**
  - Only adds edges.

- **zCall (zero call)**
  - Call instructions are disallowed.

- **zLib (zero library call)**
  - Non-executed library calls are disallowed.

- **zFun (zero functionality)**
  - Library calls with different functionalities are disallowed.
Generator

➢ **Assembler**
  - Disassembles the binary based on the expanded CFG.
  - Symbolizes basic blocks.

➢ **Instrumenter**
  - Concretizes targets of indirect calls/jumps.
  - Fixes callback function pointers.
  - Enforce allowed control-flows.

➢ **Fault handler**
  - Dumps call stacks and exits the execution.

➢ **Rewriter**
  - Compiles the instrumented assembly code to an object file.
  - Copies the code section into original binary.
  - Fixes exception handlers’ addresses in `.gcc_except_table` section.
Generator

➢ Assembler
  ○ Disassembles the binary based on the expanded CFG.
  ○ Symbolizes basic blocks.

➢ Instrumenter
  ○ Concretizes targets of indirect calls/jumps.
  ○ Fixes callback function pointers.
  ○ Enforces allowed control-flows.

➢ Fault handler
  ○ Dumps call stacks and exits the execution.

➢ Rewriter
  ○ Compiles the instrumented assembly code to an object file.
  ○ Copies the code section into original binary.
  ○ Fixes exception handlers’ addresses in `.gcc_except_table` section.
Generator

➢ Assembler
  ○ Disassembles the binary based on the expanded CFG.
  ○ Symbolizes basic blocks.

➢ Instrumenter
  ○ Concretizes targets of indirect calls/jumps.
  ○ Fixes callback function pointers.
  ○ Enforce allowed control-flows.

➢ Fault handler
  ○ Dumps call stacks and exits the execution.

➢ Rewriter
  ○ Compiles the instrumented assembly code to an object file.
  ○ Copies the code section into original binary.
  ○ Fixes exception handlers’ addresses in `.gcc_except_table` section.
Generator

➢ Assembler
  ○ Disassembles the binary based on the expanded CFG.
  ○ Symbolizes basic blocks.

➢ Instrumenter
  ○ Concretizes targets of indirect calls/jumps.
  ○ Fixes callback function pointers.
  ○ Enforce allowed control-flows.

➢ Fault handler
  ○ Dumps call stacks and exits the execution.

➢ Rewriter
  ○ Compiles the instrumented assembly code to an object file.
  ○ Copies the code section into original binary.
  ○ Fixes exception handlers’ addresses in `.gcc_except_table` section.
Code Reduction

➢ Comparing with Chisel
  ○ Basic blocks
    ■ Razor -- 78.8%, Chisel -- 83.4%
  ○ Instructions
    ■ Razor -- 61.9%, Chisel -- 85.1%
Functionality Validation

- Run the debloated binaries on the same test cases.

<table>
<thead>
<tr>
<th>Program</th>
<th># of Tests</th>
<th>Failed by Chisel</th>
<th>Failed by Razor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>W</td>
<td>I</td>
</tr>
<tr>
<td>bzip2</td>
<td>6</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>chown</td>
<td>14</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>date</td>
<td>50</td>
<td>5</td>
<td>--</td>
</tr>
<tr>
<td>grep</td>
<td>26</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>gzip</td>
<td>5</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>mkdir</td>
<td>13</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>rm</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>sort</td>
<td>112</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>tar</td>
<td>26</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>uniq</td>
<td>16</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

W: Wrong operation  
I: Infinite loop  
C: Crash  
M: Missing output
Effectiveness of Heuristics

- Run the debloated binaries on the different test cases.

**Code reduction with different heuristics**

![Bar chart showing code reduction with different heuristics](image-url)
## Security Benefits

<table>
<thead>
<tr>
<th>Program</th>
<th>CVE</th>
<th>Orig</th>
<th>Chisel</th>
<th>Razor</th>
</tr>
</thead>
<tbody>
<tr>
<td>bzip2</td>
<td>CVE-2010-0405</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CVE-2008-1372</td>
<td>✘</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CVE-2005-1260</td>
<td>✘</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>chown</td>
<td>CVE-2017-18018*</td>
<td>✔</td>
<td>✘</td>
<td>✘</td>
</tr>
<tr>
<td>date</td>
<td>CVE-2014-9471*</td>
<td>✔</td>
<td>✘</td>
<td>✔</td>
</tr>
<tr>
<td>grep</td>
<td>CVE-2015-1345*</td>
<td>✔</td>
<td>✘</td>
<td>✘</td>
</tr>
<tr>
<td></td>
<td>CVE-2012-5667</td>
<td>✘</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>gzip</td>
<td>CVE-2005-1228*</td>
<td>✔</td>
<td>✘</td>
<td>✘</td>
</tr>
<tr>
<td></td>
<td>CVE-2009-2624</td>
<td>✔</td>
<td></td>
<td>✘</td>
</tr>
<tr>
<td></td>
<td>CVE-2010-0001</td>
<td>✔</td>
<td>✘</td>
<td>✘</td>
</tr>
<tr>
<td>mkdir</td>
<td>CVE-2005-1039*</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rm</td>
<td>CVE-2015-1865*</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tar</td>
<td>CVE-2016-6321*</td>
<td>✔</td>
<td>✘</td>
<td>✔</td>
</tr>
</tbody>
</table>

- ✔️ binary is vulnerable to the CVE.
- ✘ binary is not vulnerable to the CVE.
- * CVEs with * are evaluated by Chisel.
Runtime Overhead

➢ On average, Razor introduces 1.7% slowdown.
  ○ 15.8% overhead for perlbench

![Runtime overhead for SPEC CPU 2006](image)
Real-world Software Debloating

➢ Firefox
  ○ Load top 50 Alexa websites.
  ○ Randomly pick 25 websites for debloating, and use the other 25 websites for testing.

➢ FoxitReader
  ○ Open and scroll 55 different PDF files.
  ○ Randomly pick 15 files for debloating, and use the other 40 files for testing.

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Firefox</th>
<th></th>
<th>FoxitReader</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>crash-sites</td>
<td>code-reduction</td>
<td>crash-PDFs</td>
<td>code-reduction</td>
</tr>
<tr>
<td>none</td>
<td>13</td>
<td>67.6%</td>
<td>39</td>
<td>89.8%</td>
</tr>
<tr>
<td>zCode</td>
<td>13</td>
<td>68.0%</td>
<td>10</td>
<td>89.9%</td>
</tr>
<tr>
<td>zCall</td>
<td>2</td>
<td>63.1%</td>
<td>5</td>
<td>89.4%</td>
</tr>
<tr>
<td>zLib</td>
<td>0</td>
<td>60.1%</td>
<td>0</td>
<td>87.0%</td>
</tr>
<tr>
<td>zFunc</td>
<td>0</td>
<td>60.0%</td>
<td>0</td>
<td>87.0%</td>
</tr>
</tbody>
</table>
Real-world Software Debloating

➢ Use N-fold validation approach to apply zLib heuristic on Firefox.
  ○ Split Alexa’s top 50 websites into five groups.
  ○ Select two groups (20 websites) for debloating and use the other 30 for testing.

<table>
<thead>
<tr>
<th>Group ID</th>
<th># of Failed Websites</th>
<th>Code Reduction</th>
<th>Failed Websites</th>
</tr>
</thead>
<tbody>
<tr>
<td>G01</td>
<td>1</td>
<td>59.3%</td>
<td>wordpress.com</td>
</tr>
<tr>
<td>G02</td>
<td>0</td>
<td>59.3%</td>
<td></td>
</tr>
<tr>
<td>G03</td>
<td>1</td>
<td>59.3%</td>
<td>wordpress.com</td>
</tr>
<tr>
<td>G04</td>
<td>1</td>
<td>59.3%</td>
<td>twitch.tv</td>
</tr>
<tr>
<td>G12</td>
<td>1</td>
<td>59.3%</td>
<td>wordpress.com</td>
</tr>
<tr>
<td>G13</td>
<td>1</td>
<td>59.5%</td>
<td>wordpress.com</td>
</tr>
<tr>
<td>G14</td>
<td>2</td>
<td>59.5%</td>
<td>twitch.tv, wordpress.com</td>
</tr>
<tr>
<td>G23</td>
<td>1</td>
<td>59.3%</td>
<td>twitch.tv</td>
</tr>
<tr>
<td>G24</td>
<td>1</td>
<td>59.3%</td>
<td>twitch.tv</td>
</tr>
<tr>
<td>G34</td>
<td>2</td>
<td>59.6%</td>
<td>twitch.tv, wordpress.com</td>
</tr>
</tbody>
</table>
Per-site Browser Isolation

Create minimal versions of web browsers for particular websites.

<table>
<thead>
<tr>
<th>Type</th>
<th>Website</th>
<th>Code Reduction</th>
<th>Heuristic</th>
<th>Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Banking</td>
<td>bankofamedica.com</td>
<td>69.4%</td>
<td>zCall</td>
<td>6.3%</td>
</tr>
<tr>
<td></td>
<td>chase.com</td>
<td>69.6%</td>
<td>zCall</td>
<td>6.5%</td>
</tr>
<tr>
<td></td>
<td>wells Fargo.com</td>
<td>68.8%</td>
<td>zCall</td>
<td>5.7%</td>
</tr>
<tr>
<td></td>
<td><strong>all-3</strong></td>
<td><strong>68.1%</strong></td>
<td><strong>zCall</strong></td>
<td><strong>5.0%</strong></td>
</tr>
<tr>
<td>E-commerce</td>
<td>amazon.com</td>
<td>71.4%</td>
<td>none</td>
<td>3.8%</td>
</tr>
<tr>
<td></td>
<td>ebay.com</td>
<td>70.7%</td>
<td>none</td>
<td>3.1%</td>
</tr>
<tr>
<td></td>
<td>ikea.com</td>
<td>70.6%</td>
<td>none</td>
<td>3.0%</td>
</tr>
<tr>
<td></td>
<td><strong>all-3</strong></td>
<td><strong>70.4%</strong></td>
<td>none</td>
<td><strong>2.8%</strong></td>
</tr>
<tr>
<td>Social Media</td>
<td>facebook.com</td>
<td>70.8%</td>
<td>zCall</td>
<td>7.7%</td>
</tr>
<tr>
<td></td>
<td>instagram.com</td>
<td>71.6%</td>
<td>zCall</td>
<td>8.5%</td>
</tr>
<tr>
<td></td>
<td>twitter.com</td>
<td>74.0%</td>
<td>none</td>
<td>6.4%</td>
</tr>
<tr>
<td></td>
<td><strong>all-3</strong></td>
<td><strong>71.8%</strong></td>
<td>none</td>
<td><strong>4.2%</strong></td>
</tr>
</tbody>
</table>
Summary

➢ Performs code reduction for deployed binaries.

➢ Uses heuristics to infer related code for given test cases.
Questions?