Collage of Static Analysis

Kwangkeun Yi

Seoul National University, Korea
http://ropas.snu.ac.kr/~kwang

2/26/2012 – 3/2/2012
17th Estonian Winter School in Computer Science, Palmse, Estonia
About Me/Us

We research to reduce/eliminate errors in software.

- statically: before execution, before sell/embed
- automatically: against explosive sw size
- to find bugs or verify their absence
Our Activities

- Research areas: *static analysis, abstract interpretation, programming language theory, type system, theorem proving, model checking, & whatever relevant*

- We published our works in:
  - POPL('11, '06), PLDI('12, '12), ESOP('12,'06), ICSE('11), TACAS('11), VMCAI('12, '11, '10), SAS, ISMM, OOPSLA, FSE, etc.
  - TOPLAS, TCS, JFP, SP&E, Acta Informatica, etc.

- Industrialization:
Course Outline

Collage of Static Analysis

- 0.5hr: Static Analysis Overview
- 1.5hr: Static Analysis Design Framework
- 1.0hr: Static Analysis Engineering Framework
- 1.0hr: Static Analysis of Multi-Staged Programs
Static Analysis Overview
One Challenge in SW

- How can we predict what our sw’s do?
- Can it be done automatically?

One approach to respond: static analysis
Impossible! Approximate, usefully.
Goals in Static Analysis

“SW MRI”   “SW fMRI”   “SW PET”

Or, more like “DNA diagnosis”.

Kwangkeun Yi
Collage of Static Analysis
target: full C, memory leak + buffer overrun errors
detection rate: 6/KLOC, speed: 100Loc/sec, industrialized (as of 2008)

**Bug-finder class**: unsound, non-global

Memory leak detection

<table>
<thead>
<tr>
<th>Programs</th>
<th>Size KLOC</th>
<th>Time (sec)</th>
<th>True Alarms</th>
<th>False Alarms</th>
</tr>
</thead>
<tbody>
<tr>
<td>gnuchess-5.07</td>
<td>17.8</td>
<td>9.44</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>tcl8.4.14</td>
<td>17.9</td>
<td>266.09</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>hanterm-3.1.6</td>
<td>25.6</td>
<td>13.66</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sed-4.0.8</td>
<td>26.8</td>
<td>13.68</td>
<td>29</td>
<td>31</td>
</tr>
<tr>
<td>tar-1.13</td>
<td>28.3</td>
<td>13.88</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>grep-2.5.1a</td>
<td>31.5</td>
<td>22.19</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>openssh-3.5p1</td>
<td>36.7</td>
<td>10.75</td>
<td>18</td>
<td>4</td>
</tr>
<tr>
<td>bison-2.3</td>
<td>48.4</td>
<td>48.60</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>openssh-4.3p2</td>
<td>77.3</td>
<td>177.31</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>fftw-3.1.2</td>
<td>184.0</td>
<td>15.20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>httpd-2.2.2</td>
<td>316.4</td>
<td>102.72</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>net-snmp-5.4</td>
<td>358.0</td>
<td>201.49</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>binutils-2.13.1</td>
<td>909.4</td>
<td>712.09</td>
<td>228</td>
<td>25</td>
</tr>
</tbody>
</table>
target: full C, memory leak + buffer overrun errors
detection rate: 6/KLOC, speed: 100Loc/sec, industrialized (as of 2008)

Bug-finder class: unsound, non-global
Buffer overrun detection

<table>
<thead>
<tr>
<th>Programs</th>
<th>Size</th>
<th>Time</th>
<th>True Alarms</th>
<th>False Alarms</th>
</tr>
</thead>
<tbody>
<tr>
<td>gzip-1.2.4</td>
<td>9.1</td>
<td>8.55</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>gnuchess-5.07</td>
<td>17.8</td>
<td>179.58</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>tcl8.4.14/unix</td>
<td>17.9</td>
<td>585.99</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>hanterm-3.1.6</td>
<td>25.6</td>
<td>52.25</td>
<td>34</td>
<td>1</td>
</tr>
<tr>
<td>sed-4.0.8</td>
<td>26.8</td>
<td>49.34</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>tar-1.13</td>
<td>28.3</td>
<td>57.98</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>grep-2.5.1a</td>
<td>31.5</td>
<td>47.26</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>bison-2.3</td>
<td>48.4</td>
<td>281.84</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>openssh-4.3p2</td>
<td>77.3</td>
<td>97.69</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>fftw-3.1.2</td>
<td>184.0</td>
<td>102.17</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>httpd-2.2.2</td>
<td>316.4</td>
<td>265.43</td>
<td>10</td>
<td>33</td>
</tr>
<tr>
<td>net-snmp-5.4</td>
<td>358.0</td>
<td>899.73</td>
<td>3</td>
<td>36</td>
</tr>
</tbody>
</table>
Static Analysis in Reality (3/3)

Sound, “semantically-deep”, global analyzer for 1MLoC C program: a scalability barrier knocked down (as of 2011)

Sound-&-global analyzer class:

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Baseline</th>
<th>Localize</th>
<th>Spd↑</th>
<th>Mem↓</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>Mem</td>
<td>Time</td>
<td>Mem</td>
</tr>
<tr>
<td>gzip-1.2.4a</td>
<td>7 K</td>
<td>772</td>
<td>240</td>
<td>3</td>
<td>63</td>
</tr>
<tr>
<td>bc-1.06</td>
<td>13 K</td>
<td>1,270</td>
<td>276</td>
<td>7</td>
<td>75</td>
</tr>
<tr>
<td>less-382</td>
<td>23 K</td>
<td>9,561</td>
<td>1,113</td>
<td>33</td>
<td>127</td>
</tr>
<tr>
<td>make-3.76.1</td>
<td>27 K</td>
<td>24,240</td>
<td>1,391</td>
<td>21</td>
<td>114</td>
</tr>
<tr>
<td>wget-1.9</td>
<td>35 K</td>
<td>44,092</td>
<td>2,546</td>
<td>11</td>
<td>85</td>
</tr>
<tr>
<td>a2ps-4.14</td>
<td>64 K</td>
<td>N/A</td>
<td>N/A</td>
<td>40</td>
<td>353</td>
</tr>
<tr>
<td>sendmail-8.13.6</td>
<td>130 K</td>
<td>N/A</td>
<td>N/A</td>
<td>744</td>
<td>678</td>
</tr>
<tr>
<td>nethack-3.3.0</td>
<td>211 K</td>
<td>N/A</td>
<td>N/A</td>
<td>16,373</td>
<td>5,298</td>
</tr>
<tr>
<td>emacs-22.1</td>
<td>399 K</td>
<td>N/A</td>
<td>N/A</td>
<td>37,830</td>
<td>7,795</td>
</tr>
<tr>
<td>python-2.5.1</td>
<td>435 K</td>
<td>N/A</td>
<td>N/A</td>
<td>11,039</td>
<td>5,535</td>
</tr>
<tr>
<td>linux-3.0</td>
<td>710 K</td>
<td>N/A</td>
<td>N/A</td>
<td>33,618</td>
<td>20,529</td>
</tr>
<tr>
<td>gimp-2.6</td>
<td>959 K</td>
<td>N/A</td>
<td>N/A</td>
<td>3,874</td>
<td>3,602</td>
</tr>
<tr>
<td>ghostscript-9.00</td>
<td>1,363 K</td>
<td>N/A</td>
<td>N/A</td>
<td>14,814</td>
<td>6,384</td>
</tr>
</tbody>
</table>
Static Analysis Scalability Improvement

Sound-&-global analyzer class

Collage of Static Analysis
Sparrow-detected Overrun Errors (1/3)

- in Linux Kernel 2.6.4
  625      for (minor = 0; minor < 32 && acm_table[minor]; minor++);
  ...
  713      acm_table[minor] = acm;

- in a proprietary code
  if (length >= NET_MAX_LEN)
      return API_SET_ERR_NET_INVALID_LENGTH;
  ...
  buff[length] |= (num << 4);

- in a proprietary code
  index = memmgr_get_bucket_index(block_size);
  ...
  mem_stats.pool_ptr[index] = prt

- in a proprietary code
  imi_send_to_daemon(PM_EAP, CONFIG_MODE, set_str, sizeof(set_str));
  ...
  imi_send_to_daemon(int module, int mode, char *cmd, int len)
  {
      ...
      strncpy(cmd, reply.str, len);
      cmd[len] = 0;
Sparrow-detected Leak Errors (2/3)

- **in `sed-4.0.8/regexp_internal.c`**

  ```c
  new_nexts = re_realloc (dfa->nexts, int, dfa->nodes_alloc);  
  new_indices = re_realloc (dfa->org_indices, int, dfa->nodes_alloc);  
  new_edests = re_realloc (dfa->edests, re_node_set, dfa->nodes_alloc);  
  new_eclosures = re_realloc (dfa->eclosures, re_node_set, dfa->nodes_alloc);  
  new_inveclosures = re_realloc (dfa->inveclosures, re_node_set, dfa->nodes_alloc);  
  if (BE (new_nexts == NULL || new_indices == NULL  
       || new_edests == NULL || new_eclosures == NULL  
       || new_inveclosures == NULL, 0))  
    return -1;
  ```

- **in proprietary code**

  ```c
  line = read_config_read_data (ASN_INTEGER, line,  
                              &StorageTmp->traceRouteProbeHistoryHAddrType,  
                              &tmpint);
  ...
  line = read_config_read_data (ASN_OCTET_STR, line,  
                                &StorageTmp->traceRouteProbeHistoryHAddr,  
                                &StorageTmp->traceRouteProbeHistoryHAddrLen);
  ...
  if (StorageTmp->traceRouteProbeHistoryHAddr == NULL) {
    config_perror  
      (‘invalid specification for traceRouteProbeHistoryHAddr’);  
    return SNMPERR_GENERR;
  }
  ```
in `mesa/osmesa.c` (in SPEC 2000)

276:    osmesa->gl_ctx = gl_create_context( osmesa->gl_visual );
...
287:    gl_destroy_context( osmesa->gl_ctx );

------------------
1164: GLcontext *gl_create_context( GLvisual *visual,
       GLcontext *share_list,
       void *driver_ctx )

... 1183:    ctx = (GLcontext *) calloc( 1, sizeof(GLc
... 1211:    ctx->Shared = alloc_shared_state();

---------------------
476: static struct gl_shared_state *alloc_shared
477: {
... 489:    ss->Default1D = gl_alloc_texture_object(ss,
490:    ss->Default2D = gl_alloc_texture_object(ss,
491:    ss->Default3D = gl_alloc_texture_object(ss, 0, 3);

----------------------
1257: void gl_destroy_context( GLcontext *ctx )
1258: {
... 1274:    free_shared_state( ctx, ctx->Shared );
A general method for automatic and sound approximations of sw run-time behaviors before the execution.

- applications: sw bug-finding, sw verification, sw optimization, sw management, etc
- under many names:
  - theory "abstract interpretation"
  - pl, se, veri. "type system", "model checking", "theorem proving"
  - cmplr "data-flow analysis", etc.
A general method for automatic and sound approximation of sw run-time behaviors before the execution.

- "before": statically, without running sw
- "automatic": sw analyzes sw
- "sound": all possibilities into account
- "approximation": cannot be exact
  - undecidable without approximation
- "general": for any source language and property
  - C, C++, C#, F#, Java, ML, Scala, Python, JVM, Dalvik, x86, etc.
  - "buffer overrun?", "memory leak?", "x=y at line 2?", "memory use ≤ 2K?", etc.
- One-on-one: 1 analyzer per language and property
128 × 22 + (1920 × −10) + 4

What value will it compute?

- static analysis: “an integer”
- static analysis: “an even number”
- static analysis: “a number in [−100000, 10000]”

\[x := 1; \text{repeat } x := x + 2 \text{ until readBool;}\]

What value will \(x\) have?

- static analysis: “an integer”
- static analysis: “a positive integer”
- static analysis: “an odd number”
Determine the target source:
- Java, JavaScript, C, C#, ML, Haskell, F#, Scala, Python, Dalvik, binary?

Determine the property to analyze:
- buffer overrun, null dereference, constant, class, deadlock, race, uncaught exception, unhandled case, etc.

(Design; Implementation; Test)\(^+\)
Every Static Analysis Has 3 Steps

- Set up equations about execution dynamics
  - in abstract semantic domains
  - about abstract execution dynamics
- Solve the equations
- Extract information from the solution
  - software errors, redundancies, parallelism, resource consumption, etc.
\texttt{x = readInt;}
\texttt{while (x \leq 99)}
\texttt{x++;}

What values does the \texttt{x} variable have during the execution?
x = readInt;

1:
   while (x ≤ 99)
2:
      x++;
3:
   end
4:

Capture the dynamics (semantics) by equations:

\[
\begin{align*}
    x_1 &= \left[-\infty, +\infty\right] \text{ or } x_3 \\
    x_2 &= x_1 \text{ and } \left[-\infty, 99\right] \\
    x_3 &= x_2 + 1 \\
    x_4 &= x_1 \text{ and } \left[100, +\infty\right]
\end{align*}
\]
<table>
<thead>
<tr>
<th><strong>Software</strong></th>
<th><strong>Mechanical Engineering</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>program</td>
<td>machine design</td>
</tr>
<tr>
<td>run by computer</td>
<td>run by nature</td>
</tr>
<tr>
<td>equations of executions</td>
<td>equations of executions</td>
</tr>
<tr>
<td>solving the equations</td>
<td>solving the equations</td>
</tr>
<tr>
<td>“will run as we expected”</td>
<td>“will move as we expected”</td>
</tr>
<tr>
<td>“embed in devices”</td>
<td>“build the machine”</td>
</tr>
<tr>
<td>PL &amp; Logic</td>
<td>Physics, Chemistry, &amp; XX-equations</td>
</tr>
</tbody>
</table>
How can we derive correct equations from program text?

- Does the equations capture all the execution dynamics?
- Non-obvious: pointers, heap structures, exceptions, high-order functions, typeless low-level hacks, etc.

\[ x = \text{readInt}; \]
\[ \text{while} \ (x \leq 99) \]
\[ \quad \ x++; \]
\[ \text{end} \]

\[ x_1 = \text{[} -\infty, +\infty \text{]} \text{ or } x_3 \]
\[ x_2 = x_1 \text{ and } [-\infty, 99] \]
\[ x_3 = x_2 \div 1 \]
\[ x_4 = x_1 \text{ and } [100, +\infty] \]

Is there always a solution to the derived equations?

- How do we compute the solution in a finite time?

\[ x_1 = \text{[} -\infty, +\infty \text{]} \text{ or } x_3 \]
\[ x_2 = x_1 \text{ and } [-\infty, 99] \]
\[ x_3 = x_2 \div 1 \]
\[ x_4 = x_1 \text{ and } [100, +\infty] \]