Automated Conversion of Legacy Code to Checked C

Aravind Machiry*, Hasan Touma*, Ray Chen*, Michael Hicks*
*University of California, Santa Barbara. machiry@cs.ucsb.edu
*University of Maryland and Correct Computation, Inc.
{htouma, {rchen, mwh}@cs.umd.edu

It is almost 2020, but we still have memory corruption vulnerabilities.

Let's use safe languages

Checked C
Fast
Backward compatible.

Address Sanitizer (ASan)

Can we retroactively make the legacy code safe?

What about Legacy code? Not feasible to rewrite.

Not backward compatible and need runtime changes.

Overview of our approach to automatically convert legacy code to Checked C

Evaluation

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>SLOC (KLOC)</th>
<th>Total Pointers</th>
<th>Checked Pointers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Icecast</td>
<td>Media server</td>
<td>18</td>
<td>2,218</td>
<td>1,769 (80.07%)</td>
</tr>
<tr>
<td>libarchive</td>
<td>Compression Library</td>
<td>151</td>
<td>14,637</td>
<td>11,266 (76.97%)</td>
</tr>
<tr>
<td>tiff</td>
<td>Image Utilities</td>
<td>72</td>
<td>7,120</td>
<td>5,007 (70.32%)</td>
</tr>
<tr>
<td>vsftpd</td>
<td>FTP Server</td>
<td>16</td>
<td>2,035</td>
<td>1,789 (87.81%)</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>257</td>
<td>27,010</td>
<td>18,831 (73.42%)</td>
</tr>
</tbody>
</table>

Detection Rate

Iterative refinement effectiveness

Conclusions and Future work

- Automated conversion of legacy code to Checked C is feasible and preliminary results are encouraging.
- Work being done on inferring bounds for array variables and checked regions.
- Available online: https://github.com/microsoft/checkedc-clang/tree/master/tools/checked-c-convert