Motivation

- Software Composition Analysis (SCA): automated dependency management in CI/CD
- Key tasks: discovering dependencies, checking for reachability of exploitable code, remediation

Dependency Analysis

- Determining third-party dependencies used by a project
- Dependency resolution is unspecified, difficult
- Static analysis: lockfiles
- Dynamic analysis: integrating with and querying package manager; 204% more dependencies

Reachability of Exploitable Code

- Annotate vulnerability-specific sinks ("vulnerable methods") by hand
- Check static call graph for reachability of sinks
- Problems: infeasible edges (false positives), soundy (false negatives)

Dynamic Analysis

- Instrument programs
- Execute test cases
- Check combined graph for reachable vulnerable methods

Combining Graphs

- Framework calls
- Static analysis false negatives (reflection, dynamic dispatch)
- Overhead, manual setup

Evaluation

<table>
<thead>
<tr>
<th>Project</th>
<th>Static vertices</th>
<th>Static edges</th>
<th>Dynamic vertices</th>
<th>Dynamic edges</th>
<th>Static sinks</th>
<th>Dynamic sinks</th>
</tr>
</thead>
<tbody>
<tr>
<td>helios</td>
<td>7930</td>
<td>26746</td>
<td>12287</td>
<td>39813</td>
<td>1</td>
<td>3616</td>
</tr>
<tr>
<td>immutables</td>
<td>30206</td>
<td>319934</td>
<td>398</td>
<td>874</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>java-apns</td>
<td>536</td>
<td>999</td>
<td>4240</td>
<td>8685</td>
<td>58</td>
<td>859</td>
</tr>
<tr>
<td>retrofit</td>
<td>1925</td>
<td>5269</td>
<td>7339</td>
<td>22565</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>