Abstract

Static application security testing (SAST) is a widely used technique that helps to find security vulnerabilities in program code at an early stage in the software development life-cycle. Since a few years, JavaScript is gaining more and more popularity as an implementation language for large applications. Consequently, there is a demand for SAST tools that support JavaScript.

We report briefly on our method for evaluating SAST tools for JavaScript as well as summarize the results of our analysis.

© 2014 SAP AG. All Rights Reserved. Page 2 of 8

Initial Observation and Assessment of Situation

Initial Situation:
- Increasing adoption of scripting languages (client-side and server-side, large frameworks, etc.)
- High false negative rate (in contrast to most other languages)

Market Analysis:
- Only three tools
  - commercially supported
  - with broad security scope
- Many other tools
  - specialized (e.g., only DOM-based XSS)
  - failed already on parsing our code

© 2014 SAP AG. All Rights Reserved. Page 3 of 8

Static Code Analysis at SAP

- Since 2010, mandatory for all products
- Multiple billions lines analyzed (several thousands of products/projects)
- Java:
  - Average size ca. 200 kLoC (up to several mLoC)
- Also important: SQLScript, Python, Ruby
- We also use: Perl, TCL, R, ...
- Mainly used tools:
<table>
<thead>
<tr>
<th>Language</th>
<th>Tool</th>
<th>Vendor</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABAP</td>
<td>CVA (SUN SEC)</td>
<td>SAP</td>
</tr>
<tr>
<td>C/C++</td>
<td>Covertity</td>
<td>HP</td>
</tr>
<tr>
<td>Others</td>
<td>Fortify</td>
<td>HP</td>
</tr>
</tbody>
</table>

© 2014 SAP AG. All Rights Reserved. Page 4 of 8
Evaluation and Assessment Approach

**Evaluation:**
- We used most sensitive “default” configuration (no SAP specific template/filters)
- We used the same evaluation targets
  - library of JavaScript “challenges” (own examples, test cases from IBM Research)
  - three SAP applications of different size (including one with server-side JavaScript using the XS Engine)

**Assessment:**
- Overall analysis:
  - how many findings in total reported categories
- Detailed comparison for
  - XSS-variants
  - All findings of the two topmost priorities (high)

Assessment Overview (Test Library)

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>Z</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan duration (in s)</td>
<td>246</td>
<td>246</td>
<td>1147</td>
</tr>
<tr>
<td>Findings (all)</td>
<td>111</td>
<td>118</td>
<td>242</td>
</tr>
<tr>
<td>Findings (high)</td>
<td>52</td>
<td>80</td>
<td>119</td>
</tr>
<tr>
<td>True positive</td>
<td>+++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>False negatives</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

We also tested three SAP applications
- Rather small (less than 100kLoC)
- Scalability is not a (big) problem (nightly scans are acceptable)
- Identified many aspects currently missing in test library

**Observations:**
- Only Z allows for
  - modifying existing checks
  - write own checks
- Y and Z have a better understanding of core JavaScript (they are very close)
- X and Z each have one check that reports most of the findings (false positives)
- Z includes checks for
  - use of outdated libraries (e.g., JQuery)
  - RegExp injection / RegExp DoS
- X includes checking of J2EE configurations
- Y mainly reports OWASP Top Ten

Conclusion and Outlook

- There is no good static analysis tool for JavaScript (applied) security available
- Static analyzers should be understood as frameworks (instead of off-the shelf tools)
- Frameworks and lack of modules creates as hard challenges as core JavaScript
- Good benchmark/evaluation libraries (similar to SAMATE) are needed

Response from tool vendors:
- Unsatisfactory results confirmed
- Fourth tool currently under development

And finally
- if you have questions (or want to discuss example libraries), please approach me
- want to see code examples, see my talk on Wednesday

Bibliography

- Achim D. Brucker and Uwe Sodan.
  Deploying static application security testing on a large scale.