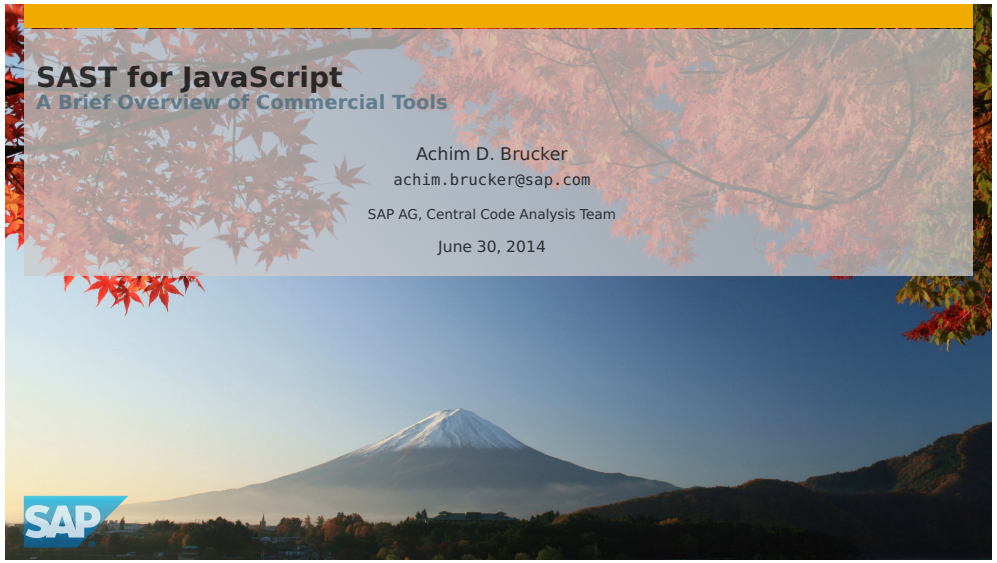


SAST for JavaScript

A Brief Overview of Commercial Tools

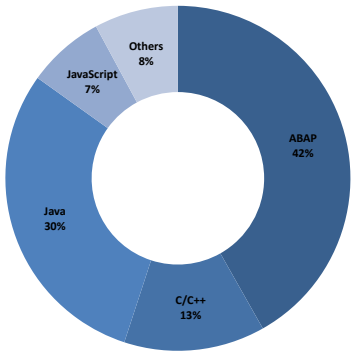
Achim D. Brucker
achim.brucker@sap.com
SAP AG, Central Code Analysis Team

June 30, 2014

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.

Static Code Analysis at SAP



Analyzed Languages (LoC) in 12/2013

- Since 2010, mandatory for all products
- Multiple billions lines analyzed
  (several thousands of products/projects)
- JavaScript:
  - Will overtake C/C++ in 2014
  - Average size ca. 200 kLoC (up to several mLoC)
- Also important: SQLScript, Python, Ruby
  We also use: Perl, TCL, R, . . .
- Mainly used tools:

Language	Tool	Vendor
ABAP	CVA (SLIN_SEC)	SAP
C/C++	Coverity	Coverity
Others	Fortify	HP

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

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)

Result Overview (Test Library)

	X	*Z*	*Y*
Scan duration (in s)	246	246	1147
Findings (all)	111	118	242
Findings (high)	52	80	119
True positive	+	+++	++
False negatives	-	+	+

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.
In Stefan Katzenbeisser, Volkmar Lotz, and Edgar Weippl, editors, *GI Sicherheit 2014*, volume 228 of *Lecture Notes in Informatics*, pages 91–101. GI, March 2014.

© 2014 SAP AG. All rights reserved