InspectJS: Leveraging Code Similarity and User-Feedback for Effective Taint Specification Inference for JavaScript

Saikat Dutta[1], Diego Garbervetsky[2], Shuvendu Lahiri[3], and Max Schäfer[4]

[1] DC/UBA. ICC/CONICET,CABA, Argentina
[2] UIUC, UrbanaUSA
[3] Microsoft Research, Seattle,USA
[4] GitHub, Oxford,UK

Abstract. Static analysis has established itself as a weapon of choice for detecting security vulnerabilities. Taint analysis in particular is a very general and powerful technique, where security policies are expressed in terms of forbidden flows, either from untrusted input sources to sensitive sinks (in integrity policies) or from sensitive sources to untrusted sinks (in confidentiality policies). The appeal of this approach is that the taint-tracking mechanism has to be implemented only once, and can then be parameterized with different taint specifications (that is, sets of sources and sinks, as well as any sanitizers that render otherwise problematic flows innocuous) to detect many different kinds of vulnerabilities.
But while techniques for implementing scalable inter-procedural static taint tracking are fairly well established, crafting taint specifications is still more of an art than a science, and in practice tends to involve a lot of manual effort.
Past work has focussed on automated techniques for inferring taint specifications for libraries either from their implementation or from the way they tend to be used in client code. Among the latter, machine learning-based approaches have shown great promise.
In this work we present our experience combining an existing machine-learning approach to mining sink specifications for JavaScript libraries with manual taint modelling in the context of GitHub's CodeQL analysis framework. We show that the machine-learning component can successfully infer many new taint sinks that either are not part of the manual modelling or are not detected due to analysis incompleteness. Moreover, we present techniques for organizing sink predictions using automated ranking and code-similarity metrics that allow an analysis engineer to efficiently sift through large numbers of predictions to identify true positives.
Published in: 2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP).
DOI: https://doi.org/10.1145/3510457.3513048

Keywords: Taint Analysis, Machine Learning, JavaScript