Dynamic Analysis Extending a Shadow Runtime for Profit

Aäron Munsters
Vrije Universiteit Brussel
Brussels, Belgium
amunster@vub.be

Angel Luis Scull Pupo
Vrije Universiteit Brussel
Brussels, Belgium
ascullpu@vub.be

Elisa Gonzalez Boix
Vrije Universiteit Brussel
Brussels, Belgium
egonzale@vub.be

CCS Concepts: • **Software and its engineering** → **Object oriented frameworks**; **Dynamic analysis**; • **Information systems** → *Web applications*; • **Security and privacy** → Information flow control.

Keywords: WebAssembly, dynamic analysis, instrumentation platform, shadow execution

1 Extended Abstract

WebAssembly [1] presents a compelling target for dynamic analysis due to its well-defined formal semantics, deterministic execution model, and cross-platform portability. Source code instrumentation platforms for WebAssembly such as Wasabi [2] and Wastrumentation [3], typically expose a set of hooks for which analysis developers can implement trap functions. These traps serve as callbacks that are invoked upon the execution of specific program events, enabling runtime introspection and intercession. However, the runtime context available within each trap remains limited. For instance, during a memory load operation, the trap function has access to the static instruction location, memory offset, and the resulting value of the operation, but it lacks information on the program state.

While these platforms offer accessible APIs that closely reflect WebAssembly's runtime semantics, they do not expose critical information for more sophisticated analyses, such as the execution value stack, global variables state, and linear memory. As a result, analyses that require tracking state across multiple traps, such as taint analysis, often resort to ad hoc reconstruction of the virtual machine's semantics. This is, however, an error-prone and labour-intensive process that must be reimplemented per analysis.

In this talk, we present ongoing work on a shadow execution framework that faithfully mimics the execution environment for WebAssembly programs designed to address these limitations by providing a reusable foundation for heavyweight dynamic analyses. We have built our prototype as an analysis layer for Wastrumentation and it is implemented in Rust.

The shadow execution framework abstracts away the manual effort of re-implementing VM semantics, enabling analyses to reason over complete execution state and increase the analysis complexity, leading to more interesting and novel insights. We demonstrate this through two extensions: (a) a taint tracking semantics that propagates metadata across instructions, and (b) an interactive online debugger that enables step-by-step inspection of program execution, even for high-level languages compiled to WebAssembly.

We aim to demonstrate how our approach shifts the burden of maintaining execution state correctness away from individual analysis implementations, leading to a shared basis for heavyweight analyses with improved correctness and potential for better performance.

Because our shadow execution is implemented in Rust, analyses built atop it can benefit from standard compiler optimizations. For example, if a specific analysis does not monitor interactions with linear memory, the Rust compiler could eliminate the corresponding instrumentation logic entirely, thereby reducing the runtime overhead and improving efficiency of the analysis.

We believe the framework also serves as a validation tool for the correctness and completeness of Wastrumentation's instrumentation logic. By executing the shadow execution in parallel with the original program, we can detect mismatches between expected and observed behavior, exposing subtle bugs in the rewriting process.

References

[1] Andreas Haas, Andreas Rossberg, Derek L. Schuff, Ben L. Titzer, Michael Holman, Dan Gohman, Luke Wagner, Alon Zakai, and JF Bastien. 2017. Bringing the web up to speed with WebAssembly. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2017)*. Association for Computing Machinery, New York, NY, USA, 185–200. doi:10.1145/3062341.3062363

[2] Daniel Lehmann and Michael Pradel. 2019. Wasabi: A Framework for Dynamically Analyzing WebAssembly. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '19)*. Association for Computing Machinery, New York, NY, USA, 1045–1058. doi:10.1145/3297858.3304068

[3] Aäron Munsters, Angel Luis Scull Pupo, and Elisa Gonzalez Boix. 2025. Wastrumentation: Portable WebAssembly Dynamic Analysis with Support for Intercession. In *39th European Conference on Object-Oriented Programming (ECOOP 2025) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 333)*, Jonathan Aldrich and Alexandra Silva (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 23:1–23:29. doi:10.4230/LIPIcs.ECOOP.2025.23