BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2026
http://excel.fit.vutbr.cz

Portable Development Environment for YARA-X

Albert Tikaiev*

Abstract
This work focuses on developing a web-based development environment and associated software components as part of the YARA-X project. While the majority of existing solutions are based on cloud architecture, this work focuses on creating a solution that does not rely on remote servers and is capable of providing similar functionality. The portability of software components was ensured through the application of WebAssembly technology. This work introduces a new language server and a linter for the YARA language, capable of operating within a dedicated web-based development environment. In conclusion, a comparison with similar tools is provided.

*xtikaia00@stud.fit.vut.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Every developer has at some point used web services designed to interact with the source code. Whether it is learning platforms like LeetCode, version control systems like GitHub, or cloud development environments like CodeSandbox, the presence of a built-in code editor is an essential part of such services. Malware researchers who use the YARA language also rely on specialized web services to manage rules written in this language. The main motivation of this work is to improve their experience by providing a web-based development environment with functionality comparable to other IDEs.

The distinguishing characteristic of the final result is that the development environment must be fully functional without a remote server. Thanks to this characteristic, the developed environment can be considered portable as it provides a simplified integration process into any web service without complicating its architecture through dependencies on external cloud services.

Currently, there are similar solutions such as the **Yara Language Server** (**YLS**) and the **Yara Development Environment** (**YDE**) [1]. These products are not portable due to the technologies used in their implementation and the adoption of cloud architecture. In addition to focusing on solving these problems, this work will also utilize new tools from the YARA-X project and ultimately serve as a replacement.

The solution described in this work considers all its software components portable in some sense. The key technology used to ensure this characteristic is **WebAssembly**. This work also considers the new development environment as a lightweight web component rather than a full-fledged web application.

The main contribution of this work is an open source language server. Currently, the language server is part of the YARA-X project [2] and is provided to users through the corresponding VS Code extension, which is shown in Figure 3.

2. Proposed design

The entire design, the relationships between the individual software components, and the distribution methods are shown in Figure 1. Although the primary goal of this project is to create a web-based development environment, the design anticipates the possibility of enhancing desktop development environments as well.

2.1 Language Server

YARA-X Language Server (**YXLS**) is a piece of software that implements the **Language Server Protocol** (**LSP**) [3]. This component is responsible for providing language features such as *Code completion*, *Go to Definition*, and others across various development environments. The portability of this component is reflected in its ability to be compiled for multiple targets, including WebAssembly.

2.2 Web-based development environment

Rule Development Environment (**RDE**) is a web component that provides an enhanced code editor. The code editor will have built-in support for the YARA language through the integration of a language server. The primary requirement for this component is that it can be easily integrated into any web page, ensuring its portability, and that it provides a programming interface for interacting with the development environment.

2.3 Linter

The newly developed linter is a tool that helps malware researchers write source code that complies with Gen's guidelines. This component will be integrated into the language server, allowing developers to receive error reports directly within their development environment.

2.4 Distribution

The language server can be integrated into code editors through plugins or extensions, whereas for web pages, a web component with an enhanced code editor is provided through the **npm** module.

3. Implementation details

3.1 Documents cache

The new language server has an important feature, which is the ability to cache documents, or more precisely, their syntax trees, something its predecessor (YLS) was unable to do. This accelerates the processing of requests that require access to symbols from included files, since they are already in memory as a syntax tree, and the language server does not need to read from disk.

3.2 Porting to the web

The porting process is complex primarily due to the limitations of the web browser environment. This section introduces these limitations and their solutions in this work.

No inter-process communication: in a native environment, the language server runs as a separate process and usually communicates with the editor via the standard input and output. But in the web environment, the spawning of child processes is not feasible, requiring a different communication mechanism. This work implements it using the **Streams API** [4] and asynchronous iterators.

Background tasks: language servers can spawn separate threads to perform background tasks, which is not possible in the same manner in a web environment. Instead, tasks can be launched with the **Web Workers API** or by enqueuing them in the event loop microtask queue, which is the approach used in the current implementation using the interface provided by **wasm-bindgen** [5].

Restricted access to the file system: despite the fact that a powerful **File System API** [6] is currently available, the implemented editor is not intended to interact with the user's local file system. Instead, it works with *virtual* files and provides a simplified programming interface for interacting with them.

3.3 Code editor

The web-based development environment is built on **Monaco Editor** [7], which also serves as the base for VS Code. The editor can be integrated into any web page simply by specifying an HTML element as its mounting point. The provided interface also supports opening documents and retrieving their content.

4. Results

As part of this work, 13 language features and 35 lint rules were developed. It has also been verified that it is possible to use a web-based development environment in frameworks such as **React** and **Vue**, as well as to bundle applications using it with **Webpack** or **Vite**.

Figure 4 shows the response time of the former language server in comparison to the new server's performance, with and without the use of caching. In all tests, language servers were required to load the contents of included files, and it is clear that caching significantly improves performance in these specific cases.

5. Conclusion

A new language server and linter were developed which could replace their predecessors, offering not only performance improvements but also new features. A web component with a development environment was also implemented, which is compatible with popular frameworks and demonstrates the portability of the language server, as well as new ways to use it.

Acknowledgements

I would like to thank my supervisor Ing. Zbyněk Křivka, Ph.D., my technical advisors Ing. Tomáš Ďuriš and Ing. Marek Milkovič for their help and guidance throughout the work. I also would like to thank Victor M. Alvarez Figure 2 for welcoming my desire to contribute to the YARA-X project.

References

[1] Matej KAŠŤÁK. Development Environment for YARA Language. Master's thesis, Brno University of Technology, Faculty of Information Technology, 2021.

[2] YARA-X. [online]. Available at: https://virustotal.github.io/yara-x/.

[3] Microsoft Corporation. Language Server Protocol. [online]. Available at: https://microsoft.github.io/language-server-protocol/.

[4] MDN Web Docs. Streams API. [online]. Available at: https://developer.mozilla.org/en-US/docs/Web/API/Streams_API.

[5] The wasm-bindgen Developers. wasm-bindgen. Facilitating high-level interactions between Wasm modules and JavaScript. [online]. Available at: https://github.com/wasm-bindgen/wasm-bindgen.

[6] MDN Web Docs. File System API. [online]. Available at: https://developer.mozilla.org/en-US/docs/Web/API/File_System_API.

[7] Microsoft Corporation. Monaco Editor. [online]. Available at: https://microsoft.github.io/monaco-editor/.