Browser Isolation via Endokernel

Aman Shanbhag
as212@rice.edu

Byron Harris
bbh3@rice.edu

Shaquille Que
stq1@rice.edu

1 Introduction

Browsers share the contradictory goals of being able to run very general workloads while also maintaining security and performance for the user. Because of the very diverse content it must be able to process and render from the Web, the browser must utilize many libraries specialized for these tasks. These libraries are often low-level in nature in order to process large volumes of data. As a result, a vulnerability in any included library may compromise the security of the browser and thus the user. Several models have been proposed to mitigate this threat. Process-level isolation can effectively ensure that different sites cannot access each other's data, but incurs a performance overhead and does not fully protect the browser from library vulnerabilities. RLBox is a framework implemented in Firefox that provides sandboxing of libraries via WebAssembly [2].

We argue that this problem is well-suited for the Endokernel [1], which allows for subprocess isolation via a nested monitor that creates two privilege levels within the process. In this paper, we will implement a minimal browser that uses the Endokernel model to enforce security and show that this is a viable way to build a browser. We will also compare this implementation to other forms of isolation.

2 Background

Modern browsers are increasingly being hardened against zero-day attacks. Doing so requires isolation of untrusted code, either from the outside or from libraries within.

Mozilla has implemented RLBox in Firefox, which is a framework that provides sandboxing via WebAssembly for libraries that Firefox uses for rendering. RLBox also support isolation via NaCl (Google Native Client) modified to support SFI and Process Isolation [2].

On the other hand, Chrome implements Site Isolation that puts each site in their own process, but this incurs the performance overhead of IPC and does not prevent the renderer from compromising the entire browser [3].

We aim to explore the space around a secure browser that does not sacrifice performance or functionality.

3 Methods and Plan

We plan to build a browser with a minimal set of features in Rust. These features include HTML and CSS parsing and rendering, image decoding, video decoding, and audio decoding, which will be implemented via external libraries. These features were chosen because they either form the core of what it means to be a browser, as in HTML rendering, or because they require the use of external libraries that are untrusted.

This browser will incorporate the Endokernel for subprocess isolation, limiting the renderer and various decoder libraries to their own subprocesses. Our browser will also allow other forms of isolation, such as via RLBox, process isolation, and Native Client in order to compare performance.

4 Milestones

1. We plan to start by developing the browser. This development will follow the chapters of Browser Engineering. However, we will many non-performance oriented features in order to minimize development time. We aim to complete this task by November 1st.

2. We will then focus on building an endokernel isolation. We aim to build this with enough abstraction that it can easily be compared to RLBox and other isolation mechanisms. The endokernel implementation will be compelted by November 14th. Implementation of other isolation techniques will be completed by November 23rd.

3. Finally, we will run the benchmarks and write everything up into a final report to be delivered on December 2nd.

References

[1] Bumjin Im, Fangfei Yang, Chia-Che Tsai, Michael LeMay, Anjo Vahldiek-Oberwagner, and Nathan Dautenhahn. The endokernel: Fast, secure, and programmable subprocess virtualization. arXiv, 2021.

[2] Shravan Narayan, Craig Disselkoen, Tal Garfinkel, Nathan Froyd, Eric Rahm, Sorin Lerner, Hovav Shacham, and Deian Stefan. Retrofitting fine grain isolation in the firefox renderer. In *Proceedings of the 29th USENIX Conference on Security Symposium*, SEC'20, USA, 2020. USENIX Association.

[3] Alexander Yip, Neha Narula, Maxwell Krohn, and Robert Morris. Privacy-preserving browser-side scripting with bflow. In *Proceedings of the 4th ACM European Conference on Computer Systems*, EuroSys '09, page 233–246, New York, NY, USA, 2009. Association for Computing Machinery.