Exploration: Memory-Safe Model of a Browser Kernel

The Chrome Security Architecture team is experimenting with building a simplified model of Chromium's process model in Rust to learn whether it can help us improve security in Chromium. The model includes core parts of how the browser manages frame trees, navigations, and assigning documents to renderer processes, which we refer to as the browser kernel. This is a much smaller set of code than Chromium's browser process, or even just `content/browser/`.

This code will live in an experimental branch and not ship to Chromium users. As it grows, it may tie into Chromium tests in the experimental branch to gain test coverage.

The initial goals of this model are:

- Provide a simpler way to reason about browser kernel behavior without the full complexity of Chromium.
- Learn from the necessary differences in representing the browser kernel in Rust vs C++, to identify possible safety improvements to make in Chromium's C++ implementation.

We are using “Crust” as a project name, referring to re-implementing part of “Chromium in Rust,” without replacing the internals, but having the ability to run it outside or alongside Chromium in tests.

Resources

- Bug: 330668787
- Experimental Branch: refs/wip/csa/exp-browser-kernel
    - https://chromium.googlesource.com/experimental/chromium/src/+log/refs/wip/csa/exp-browser-kernel
- Code location: content/browser/crust

Contributing

For Googlers, see go/chrome-experimental-branches.

Checking out the experimental branch (in your local Chromium build):

- `git remote add -f exp https://chromium.googlesource.com/experimental/chromium/src`
- `git config --local --add remote.exp.fetch '+refs/wip/csa/*:refs/remotes/exp/refs/wip/csa/*'`

Working on a CL:

- **`git fetch exp`**
- `git new-branch --upstream exp/refs/wip/csa/exp-browser-kernel local-exp-branch`
- `git fetch &&` **`git rebase`**
- `gclient sync -f -D`
- `gclient runhooks`
- `./tools/crates/run_gnrt.py vendor`
- `./tools/crates/run_gnrt.py gen`
- Make changes as usual.
- `git cl upload --no-autocc --bypass-hooks`
    - Can start by running this without --bypass-hooks to see if it catches any issues other than the code owners warning, and then add --bypass-hooks due to the code owners warning.
    - This will upload the CL pointing to the experimental branch.
    - Note: adding "-d" (dry-run) is not supported in the experimental branch and will fail with a cryptic exception.
- Once the CL is reviewed, use Gerrit to submit the CL. (Example CL.)