Web engine

Contents

Apertis provides the GTK port of WebKit as its web engine. To ensure low maintenance effort, no changes are made to the downstream branch, any improvements should go to the upstream project.

Security maintenance

Like all other major browser engines, the GTK port does not provide long term support, so security maintenance comes down to staying up to date.

The general approach Apertis takes is to follow whatever Debian provides. The project may also importing a new upstream release that has not been made available in Debian yet if an important fix is available.

Customization

Apertis has made a decision to not perform customizations to the engine with the goal of keeping maintenance efforts to a minimum. Whenever a feature is desired, it should be proposed and contributed directly upstream.

White-listing and black-listing

There has been interest in maintaining a black-list of web applications (or pages) that misbehaved. That would be for the case in which the browser gets killed because it stopped responding and the scripts watchdog was not able to restore it to working, so that those web apps or pages are not loaded automatically upon startup causing the browser to go unresponsive again.

Web[1] (codename Epiphany[2]), the GNOME web browser maintains a session file that stores information about all loaded pages, such as title, URL, and whether they are currently loading or not. If Web is quit unexpectedly, it will refuse to load any pages that were marked as still loading automatically. This same approach could be used by the Apertis web browser to not load those pages automatically or to create a blacklist database.

The white-list, on the other hand, would be used to enable applications to use lots of resources for a long time without getting killed by this infrastructure in

[1] https://wiki.gnome.org/Apps/Web
[2] https://wiki.gnome.org/Apps/Web

what could be considered a false positive. A white-list can easily be implemented, keeping a list of applications that are allowed to go over the limits should be enough.

Rendering of non-web documents

Several kinds of documents that are not strictly web documents are available on web sites for viewing and download. Some of these types of documents, such as PDFs, have become so common that some browsers embed a viewer.

WebKit itself does not have support for rendering those documents and the WebView widget provided by WebKitGTK does not support any kind of custom rendering. Applications and browsers that use the engine can also embed different widget alongside the WebView if they would like to allow viewing PDFs and other kinds of documents on the same user interface.

Scheduled and potential future work

Web runtime

There is interest in providing developers with a way to write applications using web technologies for Apertis. While this is out of the scope of this design, a small description of existing technologies and how they can be applied follows. Collabora can help in the future with a more detailed analysis of what works needs doing, specification and development of a solution.

A runner for web applications would ideally create a process for each separate application that will be executed, and use application-specific locations for storing data such as caches and the databases for the features described above – meaning it would not require any kind of special privilege, it would be a regular application. This also means permissions and resource limits can be set individually also for web applications.

The fact that more than one process would be executed does not mean a lot of memory overhead, since shared libraries are only loaded in memory once for all processes that use them. This would also have several advantages such as making managing applications permissions easier, and avoiding one application interfering with others.

Collabora believes that the best way to provide platform APIs to the web applications would be through custom bindings for relevant interfaces. That ensures the APIs will feel native to the JavaScript environment, and will significantly reduce the attack surface presented to pages, compared to a solution that makes all native libraries available.