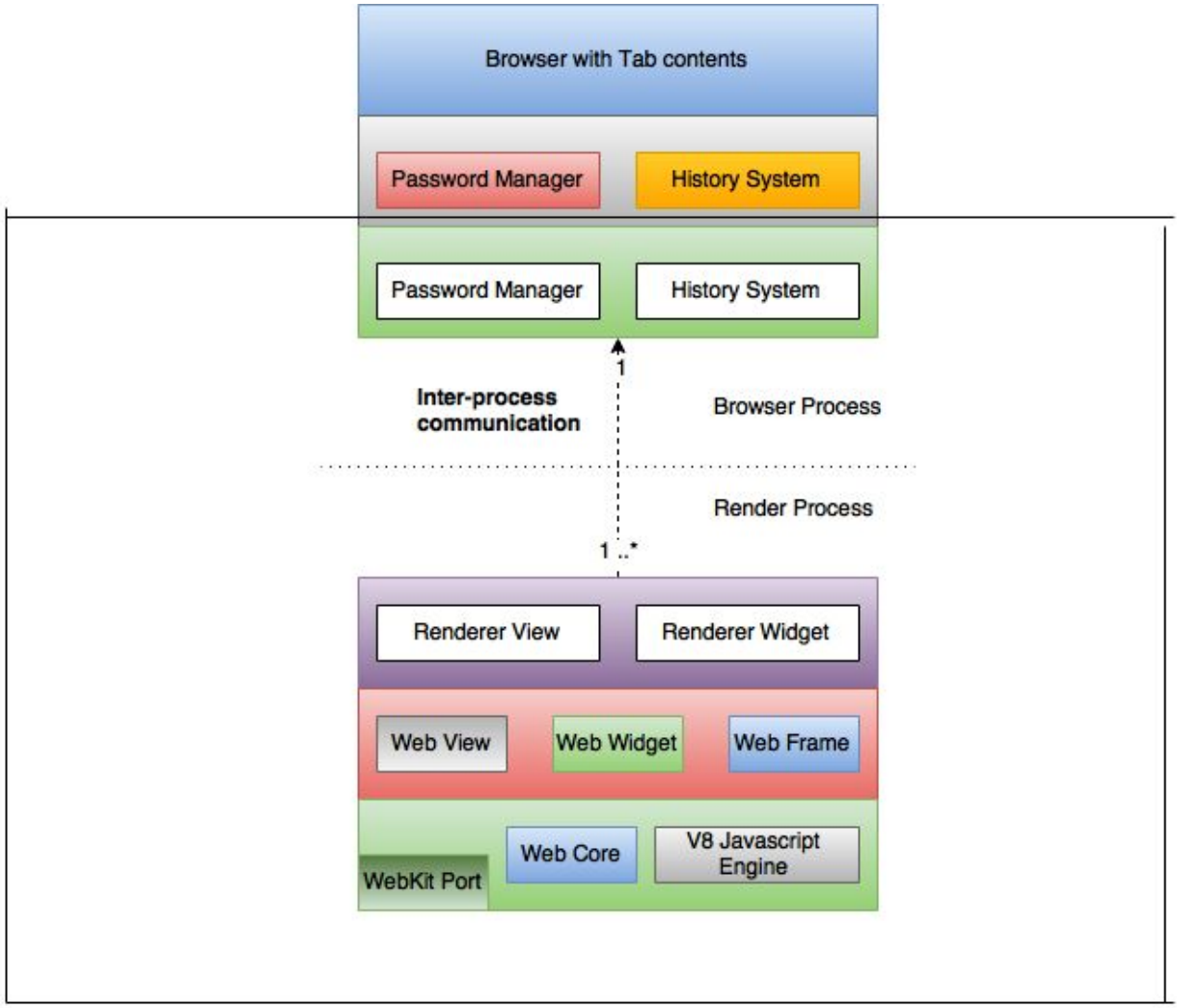
Design of Chromium

Kiran Kumar Lekkala
IS201311012

Chromium is an open-source web project from which many different variants like google chrome draws its source code from. The following is the design, functional and non-functional requirements of chromium:



Functional Requirements:

1. Basic User Interface Requirements: These are the UI interface which enables chromium to function as an user-friendly browser.

2. Multiple Processes for same Application: To make sure that the user can do many functions in the browser at the same time, there should be a provision to run multiple processes for same application.

3. Inter Process Communication: IPC is one important aspect which every browser should have so that the process can communicate with another process.

4. Secure/Safe Browsing: Encryption is one way by which the application can ensure that the user can browse safely via the browser. This is one of the most important functionality.

5. Watch Audio, Video, Images and other non-HTML features: Nowadays, we can see that the websites put dynamic content on their servers. So the browser should be able render this dynamic content like audio, video and images.

6. Run Javascript and CSS: Like the previeous feature, even the static html pages are also slowly developing into newer dynamic webpages which has graphics and animations. The browser should automatically be able to run js and css scripts.

7. Recover from Crashes: When the browsing, downloading, streaming etc. which are some of the basic utilities of the browser, are run, many a times, the browser crashes. The browser also has to ensure that it crashes minimal number of times and even if it crashes, it should get back to its normal activity.

8. Manage Downloads: Chromium should also be able to manage downloads by downloading only the content which the user wants and streaming or viewing the remaining content.

Non-Functional Requirements:

1. Security: Chromium encrypts some of the important credentials which are used in the content which is transferred from the browser to the server. Also while browsing, the browser also checks for secure HTTP certificates which the site must have. If not the browser will notify user that the connection is untrusted.

2. Performance: The browser should run in a smooth and an efficient way, promising both, functionality and speed adding better

3. Availability: The browser should work on all the available platforms which ensures that the user need not change the operating system for the browser to run.

4. Modifiability: Chromium should be able to render any website no matter what the design is for that specific website and it should load the content dynamically even if the website is changed or updated.

5. Usability: The source code should be open-source to allow developers to use and modify according to their requirements.

6. Extendibility: Chromium should be extendable by adding new features, remove old bugs. As these features get added, newer versions of chromium can be released.

Chromium Multi Process Architecture:

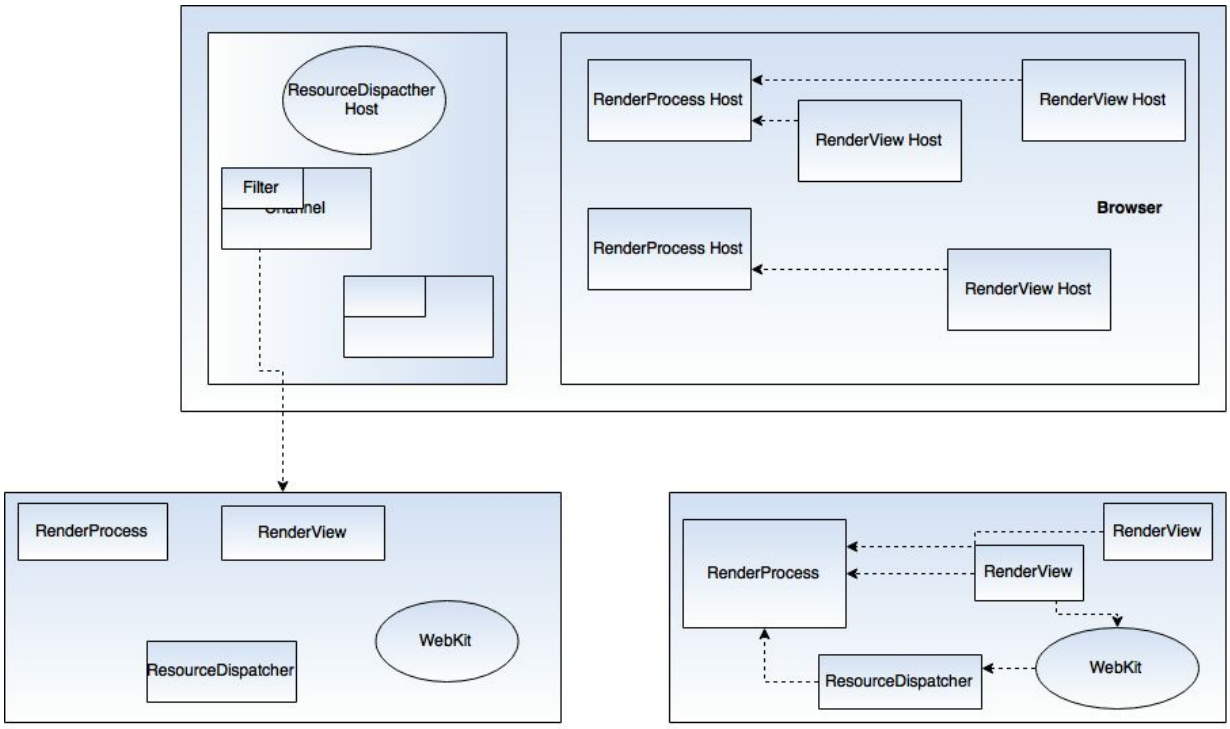


Fig. 3: Multi-process design

Chromium brings operating systems separate process architecture into the browser.

Also, it uses separate processes for browser tabs to protect the overall application from bugs and glitches in the rendering engine (i.e. A malfunction in a single tab does not affect the overall stability of the browser).

Main process that runs the UI and manages tab and plugin processes as the “browser process”. Likewise, the tab-specific processes are called “render process”.

Security Architecture of Chromium:

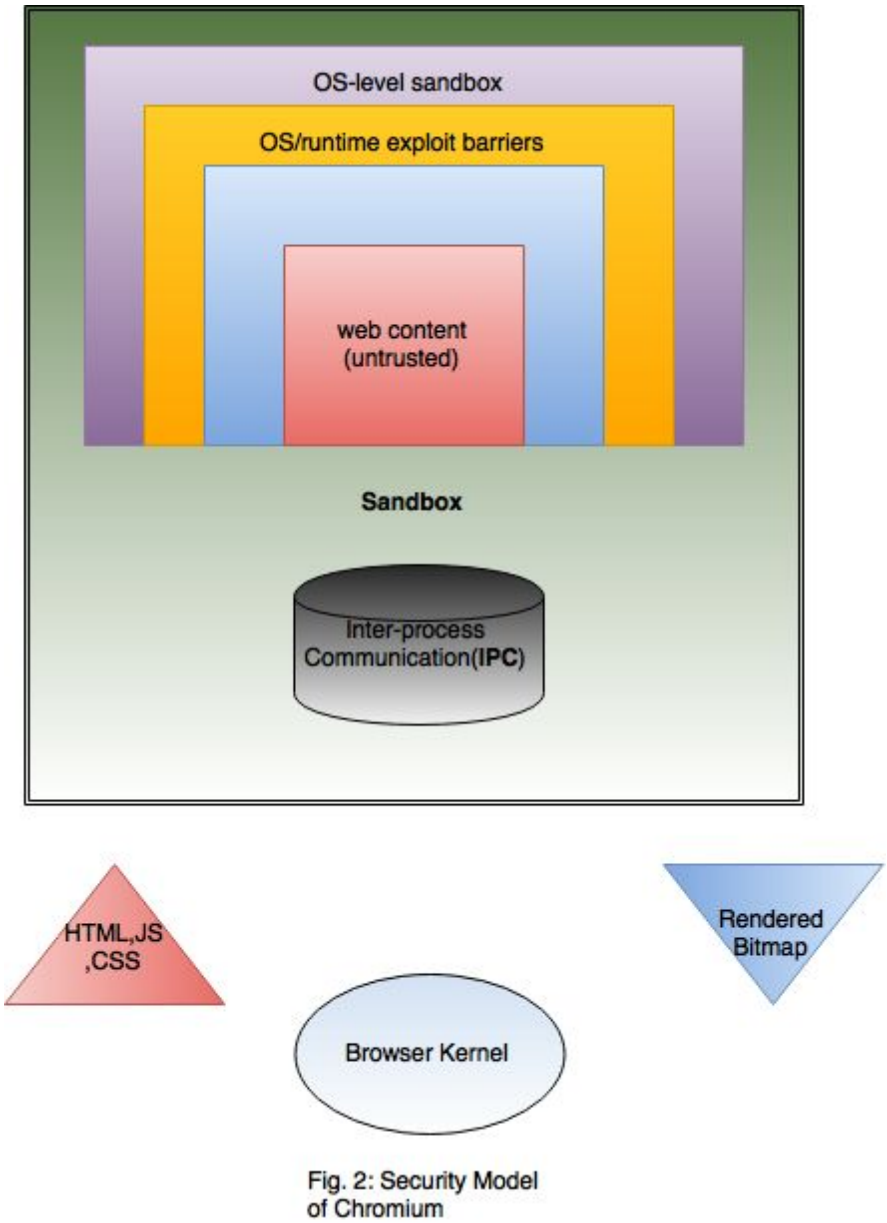


Fig. 2: Security Model of Chromium

Chromium hopes to reduce the severity of vulnerabilities by isolating a browser's complex components by adding layers of defense and reducing their privileges.

Two major components which help chromium to do so are:

1. Browser kernel (high privileged): interacting with OS, data persistence, network access. Subjected to OS level security barriers.
2. Rendering engine (low privileged): Chromium uses a modular architecture that places the complex rendering engine in a low-privilege sandbox.

Plugin architecture of Chromium:

Plugins are created by third party vendors. They cause many security vulnerabilities and instabilities. Sandboxing can not used here because they are not designed to run in Chromium's sandboxed architecture.

To compromise security, stability and usability solution is to run each plugin as separate process. Rendering engine instances will communicate with them through IPCs.

Advantages of running each plugin as a separate process:

1. Each plugin is run with users privileges. This improves usability (e.g. the Flash Player plug-in can use the user's microphone and webcam)
2. Crash of a plugin does not affect overall browser stability

Vendors could write future versions of plug-ins that operate within Chromium's sandbox, to provide greater defense against plug-in exploits.