Architecture in Practice: Chrome

Reid Holmes
NFP: Security

Security: “The protection afforded a system to preserve its integrity, availability, and confidentiality if its resources.”

Confidentiality

- Preserving the confidentiality of information means preventing unauthorized parties from accessing the information or perhaps even being aware of the existence of the information. I.e., secrecy.

Integrity

- Maintaining the integrity of information means that only authorized parties can manipulate the information and do so only in authorized ways.

Availability

- Resources are available if they are accessible by authorized parties on all appropriate occasions.
Security principles

- Security is a cross-cutting concern that cannot be retroactively added to a system.

- Several principles exist for reasoning about design decisions from a security perspective:
  - Least privilege
  - Fail-safe defaults
  - Economy of mechanism
  - Open design
  - Separation of privilege
  - Least common mechanism
  - Psychological acceptability
  - Defense in depth
Chrome

- Online content is insecure and can compromise:
  - Confidentiality: Leak user data
  - Integrity: Read/write arbitrary data on disk
  - Availability: Crash host application and/or OS

Chrome relies on least privilege, separation of privilege, and defense in depth to securely parse and render insecure content.
Chrome architecture

Chrome uses a modular architecture, which allows different components to run in different operating-system-layers. The rendering engine runs rendering Web pages. The browser kernel acts with the user’s authority and is trusted to interact with the user’s file system. The browser kernel parses this HTML, executes JavaScript, decodes images, paints to an off-screen buffer, and the user’s operating system, except for interacting with other processes.

Web Content (untrusted)

Browser Kernel (trusted)

IPC Channel

JavaScript Sandbox

OS/RunTime Exploit Barriers

OS-Level Sandbox

OS/RunTime Exploit Barriers

Other techniques include system-call rewriting (as seen in Native Client) or binary rewriting (as seen in Native Client). Mac OS X has an operating-system-provided sandbox, and Linux ports to a “low rights” mode that aims to block any desired fonts in the browser-kernel process, outside the sandbox, and the restricted Windows job. As we port Google Chrome to other platforms such as Mac and Linux, we expect to use a number of different techniques for sandboxing operating-system processes. For example, Internet Explorer 7 uses techniques for sandboxing operating-system processes. We then use a system-provided sandbox, and Linux ports to a “low rights” mode that aims to block any desired fonts in the browser-kernel process, outside the sandbox, and the restricted Windows job.

Reducing Vulnerability

Getting existing code bases such as browser developers.

To mitigate vulnerabilities in the browser itself, we use a form of a sandbox and protects different Web sites from each other. We use to limit the process itself from causing damage, even if exploits escape the JavaScript sandbox. We then use a sandbox on key security issues relevant to all users, and a low-privilege web. Even if an attacker is able to exploit vulnerabilities in the rendering engine, the sandbox will frustrate the attacker’s attempts to other resources on the user’s computer. Even if an attacker is able to exploit vulnerabilities in the rendering engine, the sandbox will frustrate the attacker’s attempts to other resources on the user’s computer.

Each of these mitigations, on its own, improves security. Taken together, the benefits multiply and help keep users safe on today’s Web.