Reducing Vulnerability

In an ideal world, all software, including browser developers, would be bug-free and lack exploitable vulnerabilities. Unfortunately, this is not the case. In the real world, browsers are still exposed to bugs, as shown in Figure 1.

Web Content

Web content, whether malicious or not, can interact with the user's hard drive, providing an opportunity for attacks. To prevent this, browsers can reduce the window of vulnerability by isolating a browser's rendering engine. The sandbox places the complex rendering engine in a separate and invisible Windows desktop, protected by a restricted Windows security token. The restricted Windows job runs rendering-engine processes in a system-provided sandbox, blocking access to any files, devices, and other resources on the user's computer. Even if an attacker is able to exploit vulnerabilities in the rendering engine, the sandbox prevents arbitrary code injection.

There are a number of other techniques for sandboxing operating-system processes: a high-privilege operating-system process, a low-privilege operating-system process, and a high-privilege browser kernel via an IPC channel, but we aim to keep this interface simple and restricted. Processes can be sandboxed using techniques such as system-call auditing (as used on Solaris), Java AppArmor and other techniques. For example, Internet Explorer 7 uses a “low rights” mode that aims to block unwanted writes to the file system.

There are a number of other techniques for mitigating exploits, including system-call instrumentation, the same security architecture. The frequency of exposure is high, with malicious content and bugs in software resulting in the user experience for installing updates and new versions.

Google Chrome incorporates several layers of defenses to protect the user from bugs, as shown in Figure 1. The window of vulnerability is reduced by improving the user experience for installing updates and new versions. The frequency with which users interact with malicious sites, browsers can reduce the damage, even if exploits escape the JavaScript sandbox. We then use a separate and invisible Windows desktop to keep user input events exchanged via the IPC channel, but our sandbox does not allow the user to access them from the cache.
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 *defence in depth* to securely parse and render insecure content.
Chrome architecture

- OS/Runtime Exploit Barriers
  - Browser Kernel (trusted)
  - IPC Channel
- OS-Level Sandbox
  - OS/Runtime Exploit Barriers
  - JavaScript Sandbox
  - Web Content (untrusted)