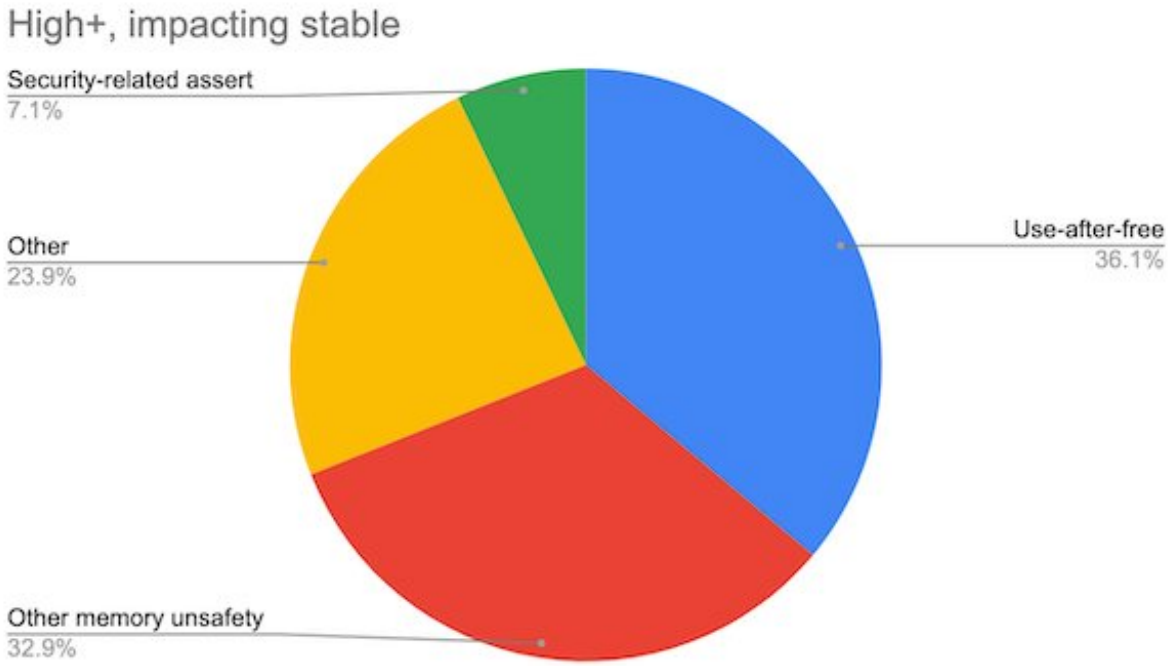
Report: Most Chrome security bugs rooted in faulty memory code

May 28 2020, by Peter Grad



Analysis based on 912 high or critical severity security bugs since 2015, affecting the Stable channel. Credit: Google

Google researchers have revealed that nearly three-quarters of all Chrome web browser security bugs stem from memory coding problems. They say their means of combatting memory management vulnerabilities through isolating browser components is reaching its maximum degree

of effectiveness and will no longer be adequate to counter future assaults.

The key factor behind the problem is Chrome's reliance on the industry standard C and C++ programming languages, neither of which was originally designed with great attention to security issues. It's understandable: the C programming language was born 48 years ago, before cyberattack was a word, years before desktop computers were commonplace, and more than a decade before the first exploitation of a vulnerability was confirmed. That first attack was the 1988 Morris worm, created by a researcher as a means to find vulnerabilities but winding up causing up to $10 million in damages.

Google engineers researching the issue examined more than 900 bugs rated "high" or "critical" dating to 2015. In just the past year alone, 130 critical bugs were linked to memory issues.

A report posted on Google's Chromium Projects site explains, "Chromium's security architecture has always been designed to assume that these bugs exist, and code is sandboxed to stop them taking over the host machine."

"That huge effort has allowed us—just—to stay ahead of the attackers," the report states. "But we are reaching the limits of sandboxing and site isolation."

Chrome is not alone in this exposure. Most of Chrome's competitors rely on C programming as well, including Microsoft Edge, Brave and Opera.

In fact, a Microsoft engineer reported a year ago the exact same number—70 percent —of his company's security issues addressed by security updates were related to memory safety. MacOS and iOS are also vulnerable to these bugs.

Firefox creator Mozilla, however, developed a new language it has been using for the past three years that was designed specifically with memory safety in mind. Google researchers say they are exploring customized C++ libraries to address these issues. They also are weighing abandoning C and C++ and switching to Rust, or other safer coding languages such as JavaScript, Swift, Kotlin or Java.

Google listed several vulnerabilities that can expose computers to malfunction or malicious activity.

- race condition: a computer erroneously attempts to perform two or more operations simultaneously that in fact must be executed in a proper sequence.
- double free: when an order to free up memory is called up more than once with the same memory address, the data structure becomes corrupted.
- use-after-free: illegal attempts to access memory after it has been freed, resulting in arbitrary code execution and exposure to unauthorized external control.
- wild pointers: uninitialized pointers aim at random addresses and cause the system to behave erratically or crash.
- buffer overflow: data exceeding permitted limits overflows into other memory buffers, corrupting or erasing data originally stored at those locations.

Google researchers say half of the 912 vulnerabilities detected were linked to "use-after-free" scenarios.

Programming code platforms developed after C and C++ have included protective measures to minimize such problems and added warning systems to alert developers to such potential conflicts.

The problem was considered serious enough that Google mandated

Chrome engineers apply "The Rule of 2" to all new browser features. Their code may not break more than two of these conditions: The code should handle untrustworthy inputs, the code should run with no sandbox, and the code should not be written in an unsafe programming language.

More information: www.chromium.org/Home/chromium … curity/memory-safety