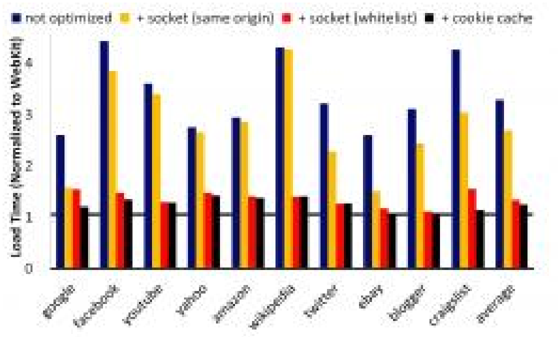
Computer scientists explore secure browser design

August 10 2012



Quark load times for the Alexa Top 10 Web sites, normalized to stock WebKit's load times. In each group the blue bar shows the unoptimized load time, the black bar shows load time in final, optimized version of Quark, and center bars show how additional optimizations improve performance.

(Phys.org) -- University of California, San Diego computer scientists explored a new approach to secure browser design in a paper presented in August 2012 at the 21st USENIX Security Symposium, the foremost research conference on computer network security. The authors are computer science professor Sorin Lerner and computer science Ph.D. students Dongseok Jang and Zachary Tatlock.

Over the past 15 years, Web-based applications have become commonplace. They are used for a wide variety of activities, from interacting with a bank or healthcare provider to managing business functions or spending leisure time interacting on a social network. As a result of this ubiquity, hackers have learned to exploit security flaws in order to access the wealth of private information tracked and accessed by Web browsers. Indeed, security breaches have become so frequent that there are now annual competitions where security experts show off their hacking chops by breaking into the latest versions of browsers, and companies such as Google pay cash prizes to people who report bugs that pose a security threat.

Because users generally want a browser that is trustworthy, secure browsing is a high priority for software developers. Yet current browsers are in fact very fragile. They are complex pieces of software with rich features that allow for flexibility and programmability, and even small bugs can make the browser vulnerable to attack. Indeed, browser vulnerabilities have been used to infiltrate the internal networks of American defense contractors and leading tech firms. Attempts to improve browser security are often ad-hoc engineering efforts; and even when formal guarantees are provided, they come in the form of proofs over a model or idealization of the browser, not the browser itself. A buggy implementation can invalidate intended guarantees and still leave users open to attack.

Jang and Tatlock, the computer science Ph.D. students, also previewed their work in a presentation to the first Center for Networked Systems (CNS) Security Day in May 2012. They explained that previous verification techniques for browser security operate on a model or abstraction of the browser, and not on its actual implementation. This has created what Tatlock calls a ‘formality gap’, a discrepancy between what is verified and what is implemented. It is through this gap that hackers can infiltrate a browser even if it has been verified using strong formal methods.

There is one known way to bridge this formality gap: implement the software in a proof assistant and use the proof assistant’s interactive environment to formally prove, in full formal detail, that the software implementation is correct. More specifically, the programmer defines a specification stating what the code should do, and then uses the proof assistant to formally prove that the code in fact satisfies this specification, beginning with the most basic axioms and then building on them. Because of the precise way in which the program has been constructed, and the foundational nature of the proof, this kind of formal verification provides extremely strong guarantees.

In the past, however, this kind of verification has faced a number of practical barriers. One of the main challenges is that building formal proofs for applications with millions of lines of code is extremely time consuming, if not completely impossible. As a result, programmers using proof assistants have either restricted themselves to verifying stripped-down versions of their applications, or have had to expand heroic effort to perform the proofs, spending much more time and programmer-power than would be practical.

Jang, Tatlock and Lerner speculated: What if there were ways to make the proofs easier by restricting the code that must be verified to a few hundred lines (as opposed to a few million)? The research team devised a technique, dubbed 'formal shim verification', for doing just this. Formal shim verification says Lerner, consists of "creating a small browser kernel which mediates access to resources for all other browser components, and then formally verifying that this browser kernel is correct in a proof assistant." In other words, only a small part of the application that is vulnerable to outside scrutiny and attack will be verified using a proof assistant.

Following these design parameters, the team created Quark, a Web browser that uses a kernel-based architecture similar to Google Chrome. In particular, the heart of Quark is a small kernel responsible for mediating access to the rest of the application. Unlike Chrome's kernel, however, the Quark kernel has been verified in full formal detail using a proof assistant, allowing it to make strong guarantees about the security of the browser. As Jang explains, Quark "exploits formal shim verification and enables us to verify security properties for a million lines of code while reasoning about only a few hundred lines of code in the kernel." This critical distinction between a verified kernel and the non-verified components allowed the researchers to incorporate a number of state-of-the-art implementations into the non-verified parts of Quark, while still maintaining strong security guarantees. For example, Quark is able to use the WebKit open-source layout engine, the same layout engine used in Safari and Chrome. Using such realistic components has made Quark into a practical and usable browser, which can successfully run complex pages like Gmail, Google Maps, Facebook and Amazon.

During the prototyping phase of Quark, one thing became clear: "When forced to choose between adding complexity to the browser kernel," says Jang, "it was always better to keep the kernel as simple as possible." As a result, the current version of Quark is in some cases too simplistic. For example, it does not support some standard features of the Web, such as third-party cookies, and in some cases it enforces non-standard security policies. Despite these current limitations, Quark is still capable of running many complex Web applications, including Facebook and Gmail.

Tatlock, Jang and Lerner are determined that this is only the first implementation of Quark. They already have some ideas about how to include a number of standard browser features without severely complicating their kernel or having to work through a fundamental redesign. Concludes Tatlock: "Our approach will handle more standard policies given design changes to our prototype and further engineering effort."

“Establishing Browser Security Guarantees through Formal Shim Verification,” by Dongseok Jang, Zachary Tatlock and Sorin Lerner at University of California, San Diego. Published in Proc. 21st USENIX Security Symposium, August 2012.

Provided by University of California - San Diego

Citation: Computer scientists explore secure browser design (2012, August 10) retrieved 23 April 2024 from https://phys.org/news/2012-08-scientists-explore-browser.html

This document is subject to copyright. Apart from any fair dealing for the purpose of private study or research, no part may be reproduced without the written permission. The content is provided for information purposes only.