

Technical Perspective

The Benefits of Capability-based Protection

By Steven D. Gribble

THANKS TO THE affordability of personal computing hardware and the usability of GUI-based PC operating systems (OSs), the vision of “a computer on every desktop and in every home” came to pass, putting the power of computing into the hands of nontechnical users. Desktop OSs were originally designed for an environment in which computers truly were personal: they were owned and used by a single user or family, and the Internet had not yet reached mainstream adoption. Users trusted the code on their computers because they installed software purchased from reputable vendors, and users did not worry about protecting the confidentiality or integrity of their data since PCs were largely single-user. As a result, desktop OSs had rudimentary or nonexistent protection frameworks, and the major threat they had to defend against was accidental damage from buggy or misconfigured software. Windows, Mac OS, and even Unix did not incorporate the strong and sophisticated protection systems that academia and industry had developed over the prior decades, since at best they were unnecessary and at worst they hurt usability.

Fortunately, personal computers did not remain personal for very long. Thanks to the Internet, users began connecting their computers to a worldwide network of billions of other computers. Small software vendors could cheaply distribute their software to a global audience, and users could use information-sharing applications, such as mail clients or P2P software, to share their files with anybody, anywhere. Web browsers and Web services have had stunning impact, facilitating the publication of and access to trillions of pages. Interestingly, Web browsers have evolved to become de facto OSs of their own, as Web sites now deliver rich, interac-


tive JavaScript-based applications to users on demand.

Unfortunately, this connectivity has brought with it a host of security headaches. Users now have to contend with virus-laden software, global-scale worms that attempt remote exploits of zero-day vulnerabilities, and spyware that attempts to capture and exfiltrate confidential information. Protection systems for desktop OSs have had to scramble to help application developers and users defend themselves against these threats. Many of the ideas, abstractions, and mechanisms developed in the context of high-security, multi-user OSs are now applicable to consumer OSs. For example, Web browsers should now be run in a sandboxed, compartmentalized environment: even if a remote Web site is able to exploit a bug in the browser, that site should not be able to gain access to confidential data from other sites the user is visiting, or harm the files and programs on the user's computer.

In the following paper, the authors describe Capsicum, their effort to bring a capability-based protection scheme to FreeBSD Unix. A capability is an unforgeable token that gives its possessor the right to access an object or resource in a computer system. Conceptually, a capability is the combination of an identifier or name for an object (such as a file name, the address of a memory region, or a remote network address) and access rights on that object (such as the ability to read the file, modify the memory region, or exchange packets with the remote host). Capabilities were first formalized in a 1966 paper by Dennis and Van Horn during MIT's exploration of multiprogramming and time-sharing OSs, and since then, many academic and industrial systems have refined and extended their ideas.

Capabilities are a great fit for sandboxing; by restricting which capabilities have been granted to an OS process, it is possible to constrain that process to access only the resources it actually requires. For example, a Web browser could be designed to launch a process for each Web page the user has open, and to use capabilities to sandbox each browser process, granting it the ability to access cached data and cookies associated with the page, but preventing it from accessing data from other sites or the user's desktop.

The major challenge the authors face is preserving Unix's existing APIs, legacy applications, and performance, while simultaneously providing a path for developers to take advantage of the strong compartmentalization and sandboxing potential of capabilities. To have impact, Capsicum needs to provide developers with an incremental adoption path; it should be easy to create sandboxes, to identify the resources to which the sandbox requires access, and to take an existing codebase and make minimal changes to it to use capabilities and sandboxed environments.

It is too early to tell whether or not Capsicum will succeed at having substantial direct impact. But, in my opinion, it has already succeeded by picking the right problem to solve: finding a way to apply the experiences and ideas developed over the past several decades of OS security research to the systems that most users actually use, in a way that minimizes the effort and expertise required from developers. 

Steven D. Gribble (gribble@cs.washington.edu) is an associate professor in the Department of Computer Science & Engineering at the University of Washington, Seattle, WA.