Making it easier to build secure Web applications

June 19 2012, by Larry Hardesty

Two years ago, a fledgling social-networking site called Blippy accidentally posted the credit card numbers of its users online. While that was a particularly egregious example, such inadvertent information leaks happen all the time: In April, for instance, the Texas attorney general's office sent civil-rights groups information on 13.1 million registered voters — including, accidentally, the complete Social Security numbers of many of them. Less than a month before, the city of Providence, R.I., accidentally released the Social Security numbers of nearly 3,000 former state employees to a local news organization.

At the USENIX Annual Technical Conference in Boston last week, MIT researchers presented a new programming system that could help prevent such inadvertent information leaks.

The system, dubbed Aeolus, is designed for programmers developing large, distributed Web applications, and it automatically keeps track of users' data-access privileges. While academics have been investigating such systems for years, Institute Professor Barbara Liskov, who led the new work, says Aeolus should prove much easier to use than its predecessors. And that, she argues, will make professional programmers more likely to adopt it.

"Just making it easy to do things like this means people will be likely to do them," says Liskov, the 2008 winner of the Turing Award, the highest honor in computer science. "On the other hand, if it's hard to do, application developers probably won't do it."

When a user of any Web application logs on, he or she generally has to supply a user name and a password. But that is by no means the last authorization check that the application performs. Every time the user invokes a new application function — moving money between bank accounts, for instance, or making a note in a patient's medical record — the application verifies that the user has the appropriate authorization. Those verifications happen behind the scenes, but the application developer still has to program them. If the developer correctly provides a thousand security checks but misses one, the application is insecure.

Hence the need for software development tools that track authorization automatically. Previous such systems, Liskov says, used what security researchers call the capability model. A capability is like an admission ticket that authorizes the holder to access certain information (except that the ticket is stored on the user's computer, and the user never knows it's there). Different functions in a Web application require their own tickets, and tickets can be passed back and forth between users. When a user attempts to invoke a function, the application simply verifies that he or she has the appropriate ticket.

Technically, such an approach is perfectly feasible, but in practice, Liskov says, it's hard for programmers to think about. The programmer doesn't have to write code for the security checks or the ticket allocations, but he or she does have to draw up rules about which users should receive which tickets on which occasions, and the ramifications of particular rule choices can be difficult to trace out.

Aeolus, which Liskov developed together with her former graduate student Winnie Cheng, current graduate students Dan Ports, David Schultz and James Cowling, and colleagues at Stanford University, Princeton University and Brandeis University, provides a much more intuitive way to think about authorization. Instead of providing rules for ticket dispensation, the programmer simply describes a hierarchy of system users. In a medical-data system, for instance, diagnostic information may be available to a patient's primary-care physician, but only contact information and appointment times are under the control of the administrative staff. A user's login credentials identify her as occupying a certain stratum of the hierarchy, and Aeolus takes care of the rest.

Among other advantages, the hierarchical approach makes it easier to revoke access privileges. A physician who changes clinics, for instance, might still need access to a statewide medical database, but should no longer be able to look at former patients' private information. With the capability model, that entails rescinding the physician's tickets; with Aeolus, on the other hand, the physician is simply demoted to a lower rung in the hierarchy.

It takes complex machinery, however, to make the system so easy to use. Aeolus includes several crucial mechanisms that work in the background to maintain security. When a user is logged into a Web application, the application needs to keep track of all the transactions the user performs. Because that record is constantly being updated, it's stored in high-speed,

easy-access memory. Aeolus ensures that none of the data in the transaction record can leak to other users of the same application or other applications running on the same server. Similarly, someone high up in the authorization hierarchy might want to use third-party software to analyze privileged data; Aeolus automatically ensures that the user's authority doesn't transfer to the software. The software can manipulate the data, but it can't leak it to the outside world.

Indeed, this raises a central point about Aeolus: It's a system for controlling information flow, not data access. That is, people at the bottom of the hierarchy can view privileged information on their computer screens; they just can't do anything else with it, such as paste it into a word-processor document — or accidentally post it online.

"The belief of people who work on this second approach, which is the information-flow approach, is that it makes it easier to build applications, compared to the access-control approach," Liskov says. Malicious users could still find ways to release private information. But, Liskov says, "the truth is that today the biggest problem is simple errors: 'I didn't really mean to do this but just did it by accident.'"

Even with Aeolus's predecessors, "the advantages in terms of strong security are evident," says Andrew Myers, a professor of computer science at Cornell University. "But it's kind of a cost-benefit analysis: For most developers, the cost in terms of changing the way you do things has been high enough that they're not willing to make that switch." With Aeolus, however, "You can specify the security policy in a much more intuitive way," Myers says. "The management of revocation is also significantly more sophisticated than in prior work."

Whether those improvements lower the cost of adoption enough that developers will start using such systems remains to be seen, Myers says. But "at some point we're going to hit the proverbial tipping point."

More information: Paper: "Abstractions for usable information flow control in Aeolus"

This story is republished courtesy of MIT News (web.mit.edu/newsoffice/), a popular site that covers news about MIT research, innovation and teaching.

Provided by Massachusetts Institute of Technology