Edward W. Felten

Webware Security

Many systems, including Java, ActiveX, JavaScript, and Web plug-ins, allow Web authors to attach an executable program to a Web page, so that anyone visiting the page automatically downloads and runs the program. These systems (collectively known as Webware) offer unique security challenges. This is not a new problem: people have always passed programs around. What is new is the scale and frequency of downloading, and the fact that it happens automatically without conscious human intervention. In one (admittedly unscientific) recent experiment, a person was found to have downloaded and run hundreds of Webware programs in a week. The same person ran only four applications from his own computer.

Simply visiting a Web page may cause you to unknowingly download and run a program written by someone you don't know or don't trust. That program must be prevented from taking malicious actions such as modifying your files or monitoring your online activities, but it must be allowed to perform its benign and useful functions. Since it is not possible (even in theory) to tell the difference between malicious and benign activity in all cases, we must accept some risk in order to get the benefits of Webware.

Despite the danger, Webware is popular because it meets a real need. People want to share documents, and they want those documents to be dynamic and interactive. They want to browse, to wander anywhere on the net and look at whatever they find.

Webware Security Models

There are two approaches to Webware security—the all-or-nothing model and the containment model. The all-or-nothing model is typified by Microsoft's ActiveX and by Netscape plug-ins. These systems rely on the user to make an all-or-nothing decision about whether to run each downloaded program. A program is either downloaded and run without any further security protection, or refused outright.

This decision can be made by exploiting digital signatures on downloaded programs. The author of a program, and anyone else who vouches that the program is well-behaved, can digitally sign it. When the program is downloaded, the user is shown a list of signers and can then decide whether to run the program.

The containment model is typified by Java from Sun Microsystems. Java allows any program to be downloaded but tries to run that program within a contained environment in which it cannot do any damage.

Problems with Both Models

Both approaches have problems. The problem with the all-or-nothing model is subtle but is impossible to fix: it puts too much burden on the user. Users are constantly bothered with questions, and they must choose between two equally unacceptable alternatives: discard the program sight unseen, or give the program free rein to damage the user's system. The all-or-nothing model causes trouble because it doesn't allow users to browse. The main problem with the containment model is its complexity. In Java, for example, there is a large security perimeter to defend, and several flaws in both design and implementation have been found, leading to the possibility of serious security breaches. Though all of the known problems have been fixed as of this writing, there is no guarantee more problems won't be found. (For a general discussion of Java security issues, see McGraw, G. and Felten, E.W. *Java Security: Hostile Applets, Holes and Antidotes*, Wiley and Sons, NY, 1997.)

Another problem with the containment model is that it is often too restrictive. Java, for example, prohibits downloaded programs from accessing files. Though this prevents malicious programs from reading or tampering with the user's private data, it also makes legitimate document-editing programs impossible. This can be addressed by making the security policy more flexible using digital signatures. When a person runs a program, their browser can verify the signatures and the person can decide whether to grant the program more privileges because of who signed it. In theory, this allows users to make finely calibrated decisions about which programs to trust for which purposes. In practice, this approach is likely to have some of the problems of the all-or-nothing model.

Still, the containment model has some advantages. Granting only a few privileges may expose the user to less risk than letting down all security barriers. And containment at least allows the system to log a program's activities.

The Challenge

Webware security is difficult because of human nature. People want to browse without worrying about security, but browsing is dangerous. Only they can decide who or what is trustworthy and how to weigh the benefits of a particular decision against the risks, but human attention to security is a precious resource that we must spend carefully. C

EDWARD FELTEN *is a professor at Princeton University.*

PAUL WATSON