# Computer Security – An End State?

Steven M. Bellovin
smb@research.att.com

It seems that one cannot open a daily paper without reading of yet another computer security breach. Worse yet, even sites that one would think would be well-protected, such as the CIA's web site, have been hacked. Is this inevitable? Will matters continue to get worse? Or is there some fix in site for the computer security problem?

Over the years, a number of fixes have been proposed. Early systems introduced the notion of a privileged state. Concepts such as the reference monitor and the trusted path have been codified. In the network sphere, some swear by encryption, while others deploy firewalls. We have even had a variety of government security standards, such as the Orange Book and the Common criteria. But computers are still being hacked. Why have these solutions not worked?

In point of fact, most security problems are caused by buggy software. Buggy software is the oldest unsolved problem in computer science, and I don't expect that to change in the forseeable future. Furthermore, the various panaceas proposed in this area—structured programming, high-level languages, formal methods, n-version programming, code walk-throughs, and more—have not succeeded. There has certainly been progress—it is no longer surprising when I find that my departmental compute server has been running continuously for six months or more—but we are still a long way from perfection. And we cannot afford 25-year shakedown periods before the complex new applications we are deploying become reliable.

Put another, we cannot have secure computer systems until we can build correct systems, and we do not know how to do that—certainly not for a long time, and probably not ever. Fred Brooks said it best in his essay *No Silver Bullets*:

> Not only are there no silver bullets now in view, the very nature of software makes it unlikely that there will be any—no inventions that will do for software productivity, reliability, and simplicity what electronics, transistors, and large-scale integration did for computer hardware.

A corollary of this is that we cannot achieve drastic improvements in computer security.

Does this mean that we are doomed? I don't think so, but we will have to adjust our attitudes, our expectations, and—of course—our professional practices.

The most important change is to realize and accept that our software *will* be buggy, *will* have holes, and *will* be insecure. Saying this is no different than saying that California *will* experience earthquakes. We don't know precisely where or when they will strike, but we know what to do to in advance: build quake-resistant structures, plan for disaster relief—and then go about our business.

We need to do the same sorts of things in the cyber world. The challenge, though, is to learn how to build hack-resistant systems. Not hack-*proof*—as I have said, that is unobtainable—but a system that

**Communications of the ACM**, vol. 44, no. 3, March 2001, pp. 131—132.

can cope with the failure, under attack, of some components. Thus, perhaps the Web site of a brokerage might be defaced, but the system architecture would be such that the account database isn't at risk. Alternatively, perhaps the account database could be compromised, but there would be sufficient backups and transaction logs that no loss of information would occur.

The second major change we must adopt is to simplify security-critical programs. In the abstract, this principle is obvious; what is less obvious is that many more programs are now security-critical. Who, ten years ago, would have thought that a word processor should be part of the trusted computing base? But there is no way to be assured of the security of such a complex component; the only possible solutions are to split off the security-sensitive pieces into small, auditable modules, or to provide new operating system primitives that will have the same effect.

There are certainly other technical approaches. For example, we can build fault-tolerant systems out of unreliable components; is there a way to do the same for security? But while that might improve the odds, it is unlikely to provide a perfect security shield. Fault-tolerant systems deal with natural failures, and Nature, and Einstein reminded us, is subtle but not malicious. Hackers, of course, do their best to shift the odds and to create improbable situations that they can exploit.

But if we succeed at this challenge—if we can build distributed systems and a cyber society that is attack-*resistant*—then our networks should survive and even flourish. No one expects major cities to be 100% crime-free, but we do expect to be able to carry out our daily activities in a reasonable degree of safety. The same can and should be true of the Net. There will never be absolute safety and perfect assurance, online or off—but there never was.

The Vandals became vandals, and descended to slashing car tires. Today they are hackers, and deface Web sites. We must ensure that they do not become Hackers and destroy our cities, or even our enjoyment of them.