Secrets and Lies

DIGITAL SECURITY
IN A NETWORKED WORLD

Bruce Schneier

WILEY COMPUTER PUBLISHING

John Wiley & Sons, Inc.
New York • Chichester • Weinheim • Brisbane • Singapore • Toronto

(An Excerpt from)

Admiral Grace Hopper said: "Life was simple before World War II. After that, we had systems. " This is an insightful comment.
Once you start conceptualizing systems, it's possible to design and build on a more complex scale. It's the difference between a building and a skyscraper, a cannon and a Patriot missile, a landing strip and an airport. Anyone can build a traffic light, but it takes a different mindset to conceive of a citywide traffic control system.
The Internet is probably the most complex system ever developed. It contains millions of computers, connected in an inconceivably complex physical network. Each computer has hundreds of software pro- grams running on it; some of these programs interact with other programs on the computer, some of them interact with other programs on other computers across the network. The system accepts user input from millions of people, sometimes all at the same time.
As the man said: "Sir, it is like a dog standing upon his hind legs, you are not surprised to see it not done well, you are surprised to see it done at all. "
Systems have several interesting properties relevant to this book. First, they are complex. Machines are simple: a hammer, a door
hinge, a steak knife. Systems are much more complicated; they have components, feedback loops, mean times between failure, infrastructure. Digital systems are daedal; even a simple computer program has hundreds of thousands of lines of computer code doing all sorts of different things. A complex computer program has thousands of components, each of which has to work by itself and in interaction with all the other components. This is why object-oriented programming was developed: to deal with the complexity of digital systems.
Second, systems interact with each other, forming even larger systems. This can happen on purpose-programmers use objects to deliberately break large systems down into smaller systems, engineers break large mechanical systems into smaller subsystems, and so on-and it can hap- pen naturally. The invention of the automobile led to the development of the modern system of roads and highways, and this in turn interacted with other systems in our daily lives to produce the suburb. The air- traffic control system interacts with the navigation systems on aircrafts, and the weather prediction system. The human body interacts with other human bodies and with the other systems on the planet. The Internet has intertwined itself with almost every major system in our society.

Third, systems have emergent properties. **In** other words, they do things that are not anticipated by the users or designers. The telephone system, for example, changed the way people interact. (Alexander Graham Bell had no idea that a telephone was a personal communications device; he thought you could use it to call ahead to warn that a telegram was coming.) Automobiles changed the way people meet, date, and fall in love. Environmental-control systems in buildings have effects on people's health, which affects the health care system. Word processing systems have changed the way people write. The Internet is full of emergent properties; think about eBay, virtual sex, collaborative authoring.
And fourth, systems have bugs. A bug is a particular kind of failure. It's an emergent property of a system, one that is not desirable. It's different from a malfunction. When something malfunctions, it no longer works properly. When something has a bug, it misbehaves in a particular way, possibly unrepeatable, and possibly unexplainable. Bugs are unique to systems. Machines can break, or fail, or not work, but only a system can have a bug.

SYSTEMS AND SECURITY

These properties all have profound effects on the security of systems. Finessing the precise definition of *secure* for now, the reason that it is so hard to secure a complex system like the Internet is, basically, because it's a complex system. Systems are hard to secure, and complex systems are that much more operose.

For computerized systems, the usual coping mechanism is to ignore the system and concentrate on the individual machines. ..the technologies. This is why we have lots of work on security technologies like cryptography, firewalls, public-key infrastructures, and tamper- resistance. These technologies are much easier to understand and to discuss, and much easier to secure. The conceit is that these technologies can mystically imbue the systems with the property of <reverence type = "hushed"> Security </reverence>.

This doesn't work, and the results can be seen in my security log from seven days of March 2000. Most of the security events can be traced to one or more of the four system properties previously listed.

Complex. The security problem with Windows 2000's Active Directory
can be directly traced to the complexity of any computer-based directory system. This is why I believe it is a design flaw; Microsoft made a design decision that facilitated usability, but hurt security.

Interactive. An interaction between the software on Intuit's Web site and
the software that DoubleClick uses to display ads to Web users resulted in information leaking from one to the other.

Emergent. According to the news story, Sony programmers had no idea
why credit card information leaked from one user to another. It just happened.

Bug Ridden. The vulnerability in Netscape Enterprise Server 3.6 was
caused by a programming bug. An attacker could exploit the bug to cause a security problem.

Many pages of this book (Part 3 in particular) are devoted to explaining in detail why security has to be thought of as a system within larger systems, but I'd like you to keep two things in mind from the beginning.

The first is the relationship between security theory and security practice. There has been a lot of work on security theory: the theory of cryptography, the theory of firewalls and intrusion detection, the theory of biometrics. Lots of systems are fielded with great theory, but fail in practice.

Yogi Berra once said, "In theory there is no difference between theory and practice. In practice there is."

Theory works best in ideal conditions and laboratory settings. A common joke from my college physics class was to "assume a spherical cow of uniform density." We could only make calculations on idealized systems; the real world was much too complicated for the theory. Digital system security is the same way: We can design idealized operating systems that are provably secure, but we can't actually build them to work securely in the real world. The real world involves design trade- offs, unseen variables, and imperfect implementations.

Real-world systems don't lend themselves to theoretical solutions; thinking they do is old-school reductionist. It only works if the spherical cow has the same emergent properties as the real Holstein. It often doesn't, and that's why scientists are not engineers.

The second thing to keep in mind is the relationship between prevention, detection, and reaction. Good security encompasses all three: a

vault to protect the lucre, alarms to detect the burglars trying to open the vault, and police that respond to the alarms and arrest the burglars. Digital security tends to rely wholly on prevention: cryptography, fire- walls, and so forth. There's generally no detection, and there's almost never any response or auditing. A prevention-only strategy only works if the prevention mechanisms are perfect; otherwise, someone will figure out how to get around them. Most of the attacks and vulnerabilities listed in this chapter were the result of bypassing the prevention mechanisms. Given this reality, detection and response are essential.