What Is Computer Security?

By Matt Bishop

Security is not an add-on or merely an operational concept, it is a property that must be designed and built into every system. Our challenge as educators is to teach our students the science underlying computer security and the techniques to synthesize and evaluate real solutions for real problems.

Computer and network security, or cybersecurity, are critical issues. But merely protecting the systems that hold data about citizens, corporations, and government agencies it is not enough. The infrastructure of networks, routers, domain name servers, and switches that glue these systems together must not fail, or computers will no longer be able to communicate accurately or reliably. Given the magnitude of securing cyberspace, a reflection on what we are trying to do seems in order. Several questions arise, such as what *exactly* the infrastructure is, what threats it must be secured against, and how protection can be provided on a cost-effective basis. But underlying all these questions is how to define a secure system.

What is security? Having it is obviously good; everyone says so. But few people define it exactly, or even nebulously. This column tries to place cybersecurity in perspective, because it is, of course, central to countries, organizations, and even home users now and in the future.

Security requirements

Consider the differences between the needs of a university and a cryptographic organization, in which foreign governments' ciphers and codes are decoded. The key difference lies in their approach to sharing information. The university fosters scholarship and open research: papers, discoveries, and work are available to the general public as well as to other academics. The cryptographic organization, on the other hand, prizes secrecy. Not only can the general public not know which ciphers have been broken, it might not even learn that ciphers *are* being broken. This sort of difference drives the need to define security.

When an organization wants to secure its systems, it must first determine what requirements to meet. The university will need to protect the integrity—and confidentiality, such as grades—of the data on its systems. It might also want to ensure that its systems are available over the Internet so students, faculty, staff, and other researchers and educators have access to information. The cryptographic organization, though, will emphasize confidentiality of all its work. Its systems should not be available over the network, because telecommuters, for examples, could download information (deliberately or accidentally) that would reside on an unsecured remote system indefinitely. Data integrity is important, but the organization would rather data be deleted than read by unauthorized people.

Security policy

Requirements dictate that some actions (and system states) be allowed and others disallowed. A *security policy,* a specific statement of what is and is not allowed, defines the system's security. If the system always stays in states that are allowed, and users can only perform actions that are allowed, the system is *secure*. If the system can enter a disallowed state, or a if user can successfully execute a disallowed action, the system is *nonsecure*.

The type of explicit definition required to design and implement security measures throughout national and international networks would somehow have to reconcile these disparate policies or, more likely, specify the services that the infrastructure could provide. Then, those who use the infrastructure could determine how to use these services to enforce their policies.

Security mechanisms

Security mechanisms enforce the policies; their goal is to ensure that the system never enters a disallowed state. The mechanisms may be technical or operational (sometimes called procedural). As an example, suppose the cryptographic organization has both unclassified and top-secret documents. Users who are not cleared to read top-secret documents cannot access them. A type of access control mechanism, called mandatory access controls, enforces this restriction. These controls are technical mechanisms.

Technical mechanisms are unsuitable for some policies. The university wants to prevent students from having music on their computers. System administrators can scan systems looking for music files, but clever students can encode the music files as text files. Music file scanners would not be able to determine that a text file was really an encoded music file. A procedural mechanism that forbids students from putting music files on their computers upon pain of suspension would be more appropriate than using such scanners.

Whether a system's set of mechanisms, taken as a whole, correctly implements the security policy is a question of *assurance*. For example, firewalls are systems that mediate network connections between a system (or set of systems on an intranet) and the Internet. The firewall can block attempts to connect to the system from the Internet. However, if the firewall software is not written correctly, the firewall might blcok connections that the security policy.

Two examples illustrate this. First, suppose the organizational policy bans the use of external peer-to-peer networks. The simplest way to enforce this policy would be to have the firewall refuse to accept messages on the relevant port. However, if the firewall is incorrectly configured, the messages will be passed on even though the policy forbids it. The protection mechanism—the firewall—is intended to enforce a security policy component. However, the mechanism is not configured properly, and so will fail in its intended task.

Second, suppose the university has a Web site for documents that are available to outside researchers. The system's security policy is that files in the Web server's data directories may be sent in reply to external requests, but no other files may be sent. The Web server program is configured to enforce this policy. Unfortunately, the server has a bug. By sending a specially crafted URL, the external request may access any file on the system. Here the mechanism fails—not because of inadequate configuration, but because of a programming error. (The ubiquitous buffer overflow error, in which the buffer is too small to hold the data copied into it, is another example of a programming error.)

Security assurances

The problem of measuring how well requirements conform to needs, policy conforms to requirements, and mechanisms implement the policy is in the realm of *assurance*. Many different methodologies help evaluate assurance. The methodology may be structured as part of the software engineering process (for high-assurance systems, this is necessary): it could test the system in particular environments to see if a policy can be violated (penetration testing falls into this category). No methodology can provide absolute assurance that a system is secure, but different methods provide different degrees of security, and the methods for evaluating assurance depend not only on the system, but also on the environment in which the evaluation occurs and on the process used to specify, design, implement, and test the system.

Assurance comes into play with people, too. How well do the system administrators understand the policies that they have to implement and enforce? Do policymakers encourage people to ask questions when aspects of the policy are not clear? Are their answers consistent? Do they ask for help when they need to understand the ramifications of the technology as they plan policies? How can the company's security staff best be organized to provide the support the policies require? Most importantly, will the staff work with people who regard security as a problem to find other ways of doing their jobs?

These questions touch on education of computer security professionals. "Education" in its broadest sense includes academic education and training. The differences between the two lie in the goals. Academic education deals primarily with principles and concepts. A good academic course will use existing technologies as examples to enhance the students' understanding, but what the student learns about how a particular technology works is ancillary to the understanding of the principles and concepts. Training, on the other hand, deals primarily with one or more technologies. Students can glean general principles and concepts from the technology, but that is secondary to understanding how the particular technology works. Which form of education is more appropriate for the specific job at hand depends on the nature of the task.

Security components

Security has three components: Requirements, policy, and mechanisms.

Requirements define security goals. They answer the question, "What do you expect security to do for you?" Policy defines the meaning of security. It answers the question, "What steps do you take to reach the goal set out above?" Mechanisms enforce policy. They answer the question, "What tools, procedures, and other ways do you use to ensure that the above steps are followed?"

These components exist in all manifestations of security.

Contrast the organizational level with the programming level. A company designing a secure operating system must first decide, secure in what sense? This is the function of company management, with the advice and assistance of other groups within the company. Once the system's requirements have been determined and the policy specified, the system must provide mechanisms adequate for enforcing that policy. These steps are integrated with the software engineering life cycle; the security is designed and implemented along with the other properties that the system must meet. Contrast this approach to adding security features to an existing system. In such cases, new features often conflict with existing mechanisms or designs. When they do, the system will not meet the security policy.

In one sense, security has a binary value: either a system is secure or it is not. However, evaluating security in this fashion evades the purpose of providing it. In general terms, a site's security strength is measured by how *closely* the requirements describe the site's needs. If the policy satisfies the requirements, and if the mechanisms enforce the policy, a company that uses a system with many security features, all of them enabled, could in fact be less secure than its competitor, which uses the same system but enables only some security features.

A classic (possibly apocryphal) example of how "too much" security can be damaging is when too many attempts to log into an account fail, thus disabling the account. One early commercial version of the Unix operating system disabled any account after three failed login attempts. An attacker broke into one of these systems and immediately tried to log in as the administrator three times, failing in each attempt. When the attacker was noticed, the system administrators tried to access the administrator account but could not—it was disabled. They had to reboot the system to reenable the account.

Sidebar: What to expect

This department has three purposes. The first is to examine security's multifaceted nature, and argue for a holistic approach in which we consider security in *all* phases of system development and deployment—specifically, in requirements analysis, system specification, design, implementation, testing, installation, and operation.

The second is validation. How can we evaluate a system's security, and how do we decide if its security is appropriate for our needs?

The third is the human side of security. Without people who know and understand the principles of security—how those principles apply in a given situation, how to define requirements and an appropriate policy, and how to use technology to implement the policy—all the efforts in the world will not create secure systems.

Among the topics we plan to explore are

- Current research areas in computer security
- Education in computer security (what is being taught, what *should* be taught, and how to bridge the gap)
- The role of research in education
- Standards, proposed standards, and standardization efforts
- Certification (its worthiness and role with respect to security professionals)
- Training (what is being taught, how applicable it is to the job, and its role in certification)

- What a security professional does on the job, and how companies structure their staffs
- Common problems facing security professionals (on the job and in education)
- Trends in employment
- Tutorials of different aspects of security problems

Our goal of is to make computer security less mysterious and better understood. Without understanding security, our attempts to protect ourselves in cyberspace are doomed to fail. We hope this department helps everyone learn what can—and cannot—be done, to protect themselves and others when working with computers and networks.

Bio for Matt Bishop

Matt Bishop is an associate professor in the Department of Computer Science at the University of California, Davis, and a co-director of the Computer Security Laboratory there. He does research in vulnerabilities analysis, the design of secure systems and software, network security, formal models of access control, and intrusion detection. He was the keynote speaker for the National Colloquia for Information System Security Education in 1997 and 2000, and is on the board of directors. His book, *Computer Security: Art and Science*, was published by Addison-Wesley in December 2002.