

Chapter 1

What is Computer Security?

The meaning of the term *computer security* has evolved in recent years. Before the problem of data security became widely publicized in the media, most people's idea of computer security focused on the physical machine. Traditionally, computer facilities have been physically protected for three reasons:

- To prevent theft of or damage to the hardware
- To prevent theft of or damage to the information
- To prevent disruption of service

Strict procedures for access to the machine room are used by most organizations, and these procedures are often an organization's only obvious computer security measures. Today, however, with pervasive remote terminal access, communications, and networking, physical measures rarely provide meaningful protection for either the information or the service; only the hardware is secure. Nonetheless, most computer facilities continue to protect their physical machine far better than they do their data, even when the value of the data is several times greater than the value of the hardware.

You probably are not reading this book to learn how to padlock your PC. *Information security* is the subject of this book. Furthermore, we are limiting our study to the insider problem: the security violations perpetrated (perhaps inadvertently) by legitimate users whom padlocks and passwords cannot deter. Most computer crimes are in fact committed by insiders, and most of the research in computer security since 1970 has been directed at the insider problem.

1.1 SECRECY, INTEGRITY, AND DENIAL OF SERVICE

Throughout this book, the discussion of computer security emphasizes the problem of protecting information from unauthorized disclosure, or information secrecy. You may find it disconcerting, as you read this book, that information integrity-protecting information from unauthorized modification or destruction-seems to be receiving no sustained attention.

There are two reasons for this seemingly one-sided point of view, one historic and one technical. First, having been funded primarily by the United States government, most computer

security endeavors have concentrated on maintaining the secrecy of classified information. This tradition has persisted even in commercial applications, where classified information is not the concern and where integrity, not secrecy, is often the primary goal. And second, the information disclosure problem is technically more interesting to computer security researchers, and the literature reflects this bias.

Fortunately, techniques to protect against information modification are almost always the same as (or a subset of) techniques to protect against information disclosure. This fact is consistently borne out in the technical measures we will discuss. In the rare cases where the techniques differ, that fact will be pointed out explicitly.

While the definition of *computer security* used in this book does, therefore, include both secrecy and integrity, the closely related area termed *denial of service* is rarely discussed here. Denial of service can be defined as a temporary reduction in system performance, a system crash requiring manual restart, or a major crash with permanent loss of data. Although reliable operation of the computer is a serious concern in most cases, denial of service has not traditionally been a topic of computer security research. As in the case of data integrity, one reason for the lack of concern is historic: secrecy has been the primary goal of government-funded security programs. But there is also an important technical reason. While great strides have been made since the early 1970s toward ensuring secrecy and integrity, little progress has been made in solving denial of service because the problem is fundamentally much harder: preventing denial of service requires ensuring the complete functional correctness of a system—something unlikely to be done in the foreseeable future.

If denial of service is your only concern, you should refer to such topics as structured development, fault tolerance, and software reliability. Most of the techniques for building secure systems, however, also help you build more robust and reliable systems. In addition, some security techniques do address certain denial-of-service problems, especially problems related to data integrity. This book will indicate when those techniques apply.

To sum up, *security* relates to secrecy first, integrity second, and denial of service a distant third. To help you remember this, memorize the computer security researcher's favorite (tongue-in-cheek) phrase: "I don't care if it works, as long as it is secure."

1.2 TRUSTED SYSTEM EVALUATION CRITERIA

The U.S. Department of Defense has developed its own definition of computer security, documented in *Trusted Computer System Evaluation Criteria* (Department of Defense 1985), also called "the Orange Book" after the color of its cover /and hereafter shortened to "the *Criteria*"). The document employs the concept of a *trusted computing base*, a combination of computer hardware and an operating system that supports untrusted applications and users. The seven levels of trust identified by the Criteria range from systems that have minimal protection features to those that provide the highest level of security modern technology can produce (table 1-1). The Criteria attempts to define objective guidelines on which to base evaluations of both commercial systems and those developed for military applications. The National Computer

Security Center, the official evaluator for the Defense Department, maintains an Evaluated Products List of commercial systems that it has rated according to the *Criteria*.

The *Criteria* is a technical document that defines many computer security concepts and provides guidelines for their implementation. It focuses primarily on general-purpose operating systems. To assist in the evaluation of networks, the National Computer Security Center has published the *Trusted Network Interpretation* (National Computer Security Center 1987), that interprets the *Criteria* from the point of view of network security. The *Trusted Network Interpretation* identifies security features not mentioned in the *Criteria* that apply to networks and individual components within networks, and shows how they fit into the *Criteria* ratings.

Class	Title	Key Features
A1	Verified Design	Formal top-level specification and verification, formal covert channel analysis, informal code correspondence demonstration
B3	Security Domains	Reference monitor (security kernel), "highly resistant to penetration"
B2	Structured Protection	Formal model, covert channels constrained, security-oriented architecture, "relatively resistant to penetration"
B1	Labeled Security Protection	Mandatory access controls, security labeling, removal of security-related flaws
C2	Controlled Access	Individual accountability, extensive auditing, add-on packages
C1	Discretionary	Discretionary access controls, protection against accidents among cooperating users
D	Minimal Protection	Unrated

Table 1-1. Trusted System Evaluation Criteria Ratings. In order for a system to be assigned a rating, it must meet all the technical requirements for its class in the four areas of security policy, accountability, assurance, and documentation. The requirements are cumulative, moving from class D to class A1.

You can be sure that a system rated high according to the *Criteria* (that is, at class A1 or B3) has been subject to intense scrutiny, because such systems are intended to protect classified military information. In order to attain such a high rating, a system has to be designed with security as its most important goal. While systems rarely qualify for any rating without some changes, most commercial operating systems can achieve a C1 or C2 level with a few enhancements or add-on packages. The Evaluated Products List is short because the *Criteria* is relatively new and evaluations take a long time. Also, many vendors have not yet shown an interest in submitting their products for evaluation.

While most of the technical concepts in the *Criteria* are covered in this book, we will pay little attention to its rating scale. If your interest is in developing a system for United States government use, the scale is important; for other applications, you will be more interested in specific features than in the ratings.

REFERENCES

Department of Defense. 1985a. DoD Trusted Computer System Evaluation Criteria. DOD 5200.28-STD. Washington, D.C.: Department of Defense. (U.S. Government Printing Office number 008-000-00461-7.)

The DoD criteria for evaluating and rating operating systems according to a scale based on security features and assurance. This document discusses many of the computer security concepts covered in this book.

National Computer Security Center. 1987. Trusted Network Interpretation. NCSC-TG-005. Ft. George G. Meade, Md.: National Computer Security Center.

An interpretation of the Trusted Computer System Evaluation Criteria for networks and network components.

Chapter 2

Why Systems Are Not Secure

Despite significant advances in the state of the art of computer security in recent years, information in computers is more vulnerable than ever. Each major technological advance in computing raises new security threats that require new security solutions, and technology moves faster than the rate at which such solutions can be developed. We would be fighting a losing battle, except that security need not be an isolated effort: there is no reason why a new technology cannot be accompanied by an integrated security strategy, where the effort to protect against new threats only requires filling in a logical piece of a well-defined architecture.

We probably cannot change the way the world works, but understanding why it works the way it does can help us avoid the typical pitfalls and choose acceptable security solutions. This chapter explores some of the classic reasons why the implementation of security lags behind its theory.

2.1 SECURITY IS FUNDAMENTALLY DIFFICULT

Why are computer systems so bad at protecting information? After all, if it is possible to build a system containing millions of lines of software (as evidenced by today's large operating systems), why is it so hard to make that software operate securely? The task of keeping one user from getting to another user's files seems simple enough—especially when the system is already able to keep track of each user and each file.

In fact, it is far easier to build a secure system than to build a correct system. But how many large operating systems are correct and bug-free? For all large systems, vendors must periodically issue new releases, each containing thousands of lines of revised code, much of which are bug fixes. No major operating system has ever worked perfectly, and no vendor of an operating system has dared offer a warranty against malfunctions. The industry seems resigned to the fact that systems will always have bugs. Yet most systems are reasonably dependable, and most of them adequately (but not perfectly) do the job for which they were designed.

What is adequate for most functions, however, is not sufficient for security. If you find an isolated bug in one function of an operating system, you can usually circumvent it, and the bug will have little effect on the other functions of the system: few bugs are fatal. But a single security “hole” can render all of the system’s security controls worthless, especially if the bug is discovered by a determined penetrator. You might be able to live in a house with a few holes in the walls, but you will not be able to keep burglars out.

As a result, securing a system has traditionally been a battle of wits: the penetrator tries to find holes, and the designer tries to close them. The designer can never be confident of having found all the holes, and the penetrator need not reveal any discoveries. Anyone entrusting sensitive information to a large operating system or to a computer on a network has reason to be concerned about the privacy of that information. If the information is valuable enough to a penetrator to warrant the effort, there is little reason to assume that the penetrator will not succeed.

But of course there is hope: with appropriate techniques, a system can be built that provides reasonably high assurance of the effectiveness of its security controls—a level of assurance much higher than that of the system’s overall correctness. The important factor is not the likelihood of a flaw (which is high), but the likelihood that a penetrator will find one (which we hope is very low). While we never can know whether a system is perfectly secure, we can build a system in a way that will make the penetrator’s job so difficult, risky, and costly that the value to the penetrator of successful penetration will not be worth the effort.

The key to achieving an acceptable degree of security is the systematic use of proper techniques. Ad hoc security measures provide, at best, insignificantly increased protection that rarely justifies their expense. At worst, they provide a false sense of security that renders the users more susceptible than ever to the real threats.

2.2 SECURITY IS AN AFTERTHOUGHT

Despite the publicity about computer security in the press, computer and software vendors have rarely taken the trouble to incorporate meaningful security measures into their systems. Security, if considered at all, usually comes at the bottom of a list that looks something like this:

Functions: What does it do?

Price: What does it cost?

Performance: How fast does it run?

Compatibility: Does it work with earlier products?

Reliability: Will it perform its intended function?

Human Interface: How easy is it to use?

Availability: How often will it break?

-
-
-

Security Functions: What protection features does it provide?

Security Assurance: How foolproof are the protection features?

Based on past and current practice, you might say that this entire book is about two of the least important factors in the design of computer systems.

It is unfair to fault vendors entirely for this lack of attention to security. While customers may want improved security, they usually have second thoughts when security features adversely affect other, “more important” features. Since few customers are willing to pay extra for security, vendors have had little incentive to invest in extensive security enhancements.

A few vendors have taken steps to help the few security-conscious customers who are willing to invest in additional protection. These customers include not only the government but some banks, manufacturers, and universities. Several add-on security packages for major operating systems have been on the market for some time. The most notable of these are CGA Software Products Group’s TOP SECRET, Uccel Corporation’s ACF2, and IBM’s RACF, all for IBM’s MVS operating system. Stronger mandatory controls (a subject of chapter 6) designed to be integrated into the operating system appear in SES/VMS, an enhancement to VMS offered by Digital Equipment (Blotky, Lynch, and Lipner 1986), and are under development in the Sperry (now Unisys) 1100 operating system (Ashland 1985). These packages and enhancements are commercially viable despite their significant purchase and administrative costs. Several vendors have made a considerable investment in internal security enhancements to their operating systems without cost add-ons. These systems include DEC’s VMS and Honeywell’s Multics (Organick 1972; Whitmore et al. 1973). Control Data has also incorporated security enhancements into its NOS operating system. Honeywell was the first to offer commercially a highly secure minicomputer, the SCOMP (Fraim 1983), based on a security kernel, (a subject of chapter 10). Gemini Computers offers the GEMSOS operating system, also based on a security kernel (Schell, Tao, and Heckman 1985).

These and several other examples show that there has always been a certain demand for security features in the user community. But the examples also show that demand is fairly weak and can easily evaporate if the features should have an adverse impact on cost or any other functions.

2.3 SECURITY IS AN IMPEDIMENT

A common perception among users is that security is a nuisance. Security measures are supposed to thwart someone who tries to break the rules; but because of poorly integrated ad hoc solutions, security measures often interfere with an honest user’s normal job.

Vendors often implement security enhancements in response to specific customer demands. Such enhancements, made to existing systems at minimal cost, often result in reduced convenience or poor performance. Vendors commonly adopt the attitude that a customer who wants security badly enough should be willing to live with the inconvenience.

Many customers take it upon themselves to fix security problems at their own sites. Because of inherent limitations in the system, fixing security problems often requires restrictive procedural controls: limited access from remote terminals; restricted physical access to local

terminals, and printers; multiple passwords or logins; frequent password changes; automatic disconnect after periods of inactivity; and call-back devices. Many of these controls do not substantially increase the security of the system, but they do foster the notion that security is painful. Because users and managers do not see a way around the inconveniences, security is often employed only as a last resort, when a problem has already occurred or a clear threat exists.

2.4 FALSE SOLUTIONS IMPEDE PROGRESS

The computer industry, like other industries, is subject to fads. Fads in the computer security area can have a serious negative effect on the overall progress toward achieving good security, because progress stops when people think they have the answer. Since few people have a good understanding of security, security fixes are particularly subject to snake-oil salesmanship.

One misconception (fortunately short-lived) involved *data encryption*; that is, encoding information using a password or secret key so that it cannot be deciphered by unauthorized individuals. Data encryption is indispensable for communications and is useful for protecting the media used to store files, but it does not address the general computer security problem. Few of the penetration techniques used by various “tiger teams” charged with finding security holes in systems would be thwarted by encryption. The primary problem with file encryption is that it does nothing to increase the level of trust in the operating system; and if you do not trust your operating system to protect your files, you cannot trust it to encrypt your files at all the right times or to protect the encryption keys properly. Nonetheless, simplistic statements are still occasionally encountered that claim that securing an operating system is unnecessary if all the files are encrypted. Section 13.2 discusses the legitimate role of encryption in communications and the relationship of encryption to computer security.

A popular security device is the *call-back modem*. The idea is that you telephone a computer from your home or office terminal and identify yourself (via a password) to the modem on the remote computer through your terminal. The computer’s modem verifies that the password is correct and tells you to hang up. The modem then looks up your home telephone number in a list, and calls you back. Nobody can dial into the system and masquerade as you, even if that person knows your password, unless that person also uses your phone. Call-back devices are attractive because they do not require any modification to the system being protected—a classic example of add-on security. The danger in these devices is the risk of being lulled into complacency because you feel that only “good guys” can get to your system. You may decide that it is never necessary to change passwords or to enforce any control over the types of passwords people use. You may become lax about access control within your system, allowing too many of your users access to too much information. You may forget that half of your security problem is a matter of keeping your users isolated from each other—not keeping outsiders out.

The worst problem with call-back modems, however, is that they may cause you to forget that there are other ways people can get into your system. Does your system have a connection to a commercial network from which users can log in? Can you trust all other systems with which your system communicates? If one of your users accesses your system via a modem on a personal computer, how do you ensure that the personal computer has not been penetrated by an outsider via that modem? Considering the problems that call-back modems cannot solve and

weighing the cost of these devices against simple measures such as better password control, it is hard to see their value.¹

An example involving the use of passwords shows how a security feature intended for one application can be applied inappropriately to another. Because passwords are so good at controlling a user's access to the system, they are often used for other types of access control access to certain applications in a system, access to certain files, or freedom to carry out certain operations. Password schemes are attractive because they are so easy to implement and to add onto existing systems.

But passwords are inappropriate for many of these applications, especially when a single password is issued to several people (for access to a common file, for example. When one person in the group leaves the company, the password must be changed and the new password manually distributed. If a break-in by an insider occurs, it is impossible to tell who is at fault. And the greater the number of people who know the password, the greater the chance that it will be revealed accidentally.

Another misuse of passwords involves the requirement on some systems that the user at a terminal reenter the password periodically—supposedly to ensure that the intended user and not an intruder is at the terminal. This feature is dangerous for two reasons. First, repeated entry of the password greatly increases the risk that someone will be looking over the user's shoulder when the password is entered. Second, the prompt for a password, appearing at unexpected times during a session, is highly susceptible to spoofing by a Trojan horse (see chapter 7). Section 6.2.1 lists additional ways in which passwords may be misused.

The false sense of security created by inappropriate use of passwords weakens the impetus to seek better controls. The danger of using such ad hoc solutions to address isolated problems is that one can lose sight of the fundamental problems.

2.5 THE PROBLEM IS PEOPLE, NOT COMPUTERS

Many organizations believe that computer security technology is irrelevant to real-world problems because nearly all recorded cases of computer abuse and fraud are non-technical. Computer crime usually involves exploitation of weaknesses in procedural or personnel controls, not weaknesses in internal controls. Hence, as long as relatively easy, non-technical ways exist to commit a crime, technical controls will be viewed as superfluous.

But these organizations often fail to recognize that the computer can protect against flawed procedural controls. As we shall discuss in section 3.1, technical controls can often be used to ease the burden of procedural controls. It is distressing, for example, to hear claims that attacks by former employees represent personnel problems that the computer cannot solve, when the system can easily be instrumented to defend itself against this threat.

¹The idiosyncrasies of the telephone system provide a number of additional ways to defeat most call-back devices, but that is another story.

Consider, too, what will happen when procedural controls are strengthened to the point that technical penetration becomes the path of least resistance. Since many years are needed to make major security improvements to existing systems, a sudden explosion of technical crimes will be very difficult to counter.

Probably because the computer industry is still in its infancy, sufficient knowledge of computers to exploit technical flaws seems to be rare among the dishonest. (On the other hand, perhaps they are so clever that they are not detected.) But as knowledge of computers becomes more common, we cannot assume that only a few honest citizens will possess the requisite skills to commit a major crime. Given the low risk of getting caught and the potentially high payoff, sophisticated computer crime is likely to become more attractive in the future, especially if the non-technical avenues to crime are sufficiently restricted.

One of the primary arguments that computers cannot prevent most cases of abuse is based on the observation that computer crimes committed by insiders usually do not involve a violation of internal security controls: the perpetrator simply misuses information to which he or she normally has access during the course of normal work responsibilities. Something akin to artificial intelligence would be required to detect such abuse automatically. But on closer inspection, we often find that people routinely gain access to more information than they need, either because the system's security controls do not provide adequately fine-grained protection or because implementing such protection within the architectural constraints of the system is too inconvenient or costly. The problem appears to be solely one of people, but it is exacerbated by a technical deficiency of the system. The technical solutions are not apparent because an organization's way of doing business is often influenced by the design (and limitations) of its computer system.

2.6 TECHNOLOGY IS OVERSOLD

There has long been the perception that true computer security can never be achieved in practice, so any effort is doomed to failure. This perception is due, in large part, to the bad press that a number of prominent government-funded secure computer development programs have received. The reasons for the supposed failure of these developments are varied:

- Programs originally intended for research have been wrongly criticized for not fulfilling needs of production systems.
- Vying for scarce funding, researchers and developers often promise more than they can deliver.
- Funding for the programs has been unpredictable, and requirements may change as the programs are shuffled among agencies. Often the requirements ultimately expressed are inconsistent with the original goals of the program, leading to unfortunate design compromises.
- Developments are often targeted to a specific model of computer or operating system, and inconsistent levels of funding have stretched out programs to the point where the original target system is technologically obsolete by the time the program is ready for implementation.

- The public does not realize that the first version of an operating system always performs poorly, requiring significant additional design and tuning before becoming acceptable. Vendors do not release such preliminary systems, postponing their “Version 1.0” announcement until the performance problems have been addressed. Government programs are highly visible, and any problems (even in early versions) tend to be viewed by critics as inherent characteristics. Worse, contracts are often written in such a way that the first version is the final product, and additional money is rarely available for performance tuning.
- Several large government procurements have specified the use of security technology that was thought to be practical at the time but was in fact based on research still in the laboratory. When the research failed to progress fast enough to satisfy the needs of the program, security requirements were waived and the program lost its credibility. Industry has understood for a long time that developing a new operating system involves far more than a one-time expense to build it; rather, a high level of continuous support is required over the life of the system. The federal government seems to have realized this, as well. Not able to commit to open-ended support, the government has largely ceased direct funding for secure operating system development, concentrating instead on specific applications and various seed efforts. A few commercial vendors are now undertaking to fill the void.

REFERENCES

- Ashland, R. E. 1985. “B1 Security for Sperry 1100 Operating System.” In *Proceedings of the 8th National Computer Security Conference*, pp. 105–7. Gaithersburg, Md.: National Bureau of Standards.
A description of mandatory controls proposed for Sperry (now Unisys) operating systems.
- Blotcky, S.; Lynch, K.; and Lipner, S. 1986. “SE/VMS: Implementing Mandatory Security in VAX/VMS.” In *Proceedings of the 9th National Computer Security Conference*, pp. 47–54. Gaithersburg, Md.: National Bureau of Standards.
A description of the security enhancements offered by Digital Equipment to upgrade security on its VMS operating system.
- Fraim, L. J. 1983. “SCOMP: A Solution to the Multilevel Security Problem.” *Computer* 16(7): 26–34. Reprinted in *Advances in Computer System Security*, vol. 2, ed. R. Turn, pp. 185–92. Dedham, Mass.: Artech House (1984).
A minicomputer-based security kernel with sophisticated hardware protection; this system is a Honeywell product.
- Organick, E. I. 1972. *The Multics System: An Examination of Its Structure*. Cambridge, Mass.: MIT Press.
A description of Multics—at that time implemented on a processor without hardware-supported protection rings.
- Schell, R. R.; Tao, T. F.; and Heckman, M. 1985. “Designing the GEMSOS Security Kernel for Security and Performance.” In *Proceedings of the 8th National Computer Security Conference*, pp. 108–19. Gaithersburg, Md.: National Bureau of Standards.
A description of a security kernel for the Intel iAPX 286 microprocessor offered by Gemini Computers.

Whitmore, J.; Bensoussan, A.; Green, P.; Hunt, D.; Kobziar, A.; and Stern, J. 1973. "Design for Multics Security Enhancements." ESD-TR-74-176. Hanscom AFB, Mass.: Air Force Electronic Systems Division. (Also available through National Technical Information Service, Springfield, Va., NTIS AD-A030801.)

A description of the enhancements incorporated into Multics to support mandatory security controls.