

## Essay 3

# A Philosophy of Security Management

David Bailey

---

It is not possible to maintain security in a system or network of more than minimum complexity unless appropriate management is possible. This essay discusses security management of complex systems, including the scope of the security manager's role and the conflicting pressures that must be balanced. It ends by discussing a strategy for the security manager that has been used successfully on a large local network.

### **The security manager's world**

Security management is an important part of the whole secure system picture. A system whose security cannot be managed is not secure, no matter what evaluators may tell you about its internal controls. Today, the security of "secure systems" is difficult to manage, and the situation is rapidly getting worse for complex computer networks and systems. This requires the security manager to adopt a strategic point of view and to place more faith in architectural and environmental controls than in controls built into operating systems.

This essay is about developing a strategy for security management. It does not recommend specific controls, specific implementation techniques, or specific products. It does not even try to describe specific elements that should be included in every policy. None of this could be done without having a great deal of information about the system in which they were to be installed. Security management is a discipline in which making trade-offs is a continual activity: controls in the system versus controls in the environment, security control versus customer convenience and productivity, strong controls versus implementation and administrative costs, and so on. This essay is about how to make those trade-off decisions.

The reader should not expect to find a recipe for protecting a system or network in this essay. Instead, the essay describes an approach to the security management job that will allow the manager to avoid “falling behind,” even if the system or network is large and complex. The observations below are based on a dozen years’ security management experience with a large and complex network used for scientific computation and falling under US government rules for handling classified data. This may not be the type of network you are trying to manage. Nevertheless, whether your network is government, commercial, or military — or large or small — there are some commonalities in the pressures on the security manager. Elements of the strategy below may be helpful.

In the rest of this essay, we take a broad view of a security manager’s responsibilities and develop a philosophy that can help one avoid the pitfalls produced by rapidly changing technology. There are a number of “rules of thumb” that apply to the world of computing in the 1990s and guide the security manager’s strategy. We begin by discussing the security manager’s role in an enterprise. Next we discuss some of the factors that affect how security management works, and finally we discuss a strategy for approaching the job.

For simplicity, only computer networks are discussed below. Every modern computing system can be thought of in network terms, so if we deal with network management issues, stand-alone systems will also have been addressed.

### **The basic job description**

Security management is not simply watching over the computer and its operation after it has been installed (although this is an important activity). It also includes tasks that must be completed before installation, such as setting and articulating the security policy. It even includes tasks that are not started until after the system is removed from service, such as sanitization. These tasks can be summarized as two basic responsibilities:

- The security manager has an obligation to management to ensure that the security requirements imposed on the system will adequately protect the organization’s resources and data.
- The security manager has an obligation to management to ensure that the system is operated in a manner that satisfies its security requirements and to report significant or continuing deviations in security.

Every system gets much of its security “direction” from the outside. Government and military systems must satisfy an agency security policy. Banking and insurance systems must satisfy various laws and rules im-

posed by regulators. Other commercial systems must satisfy a variety of laws and decisions imposed by upper management or the board of directors. However, external sources of direction, even if they are reasonably specific, do not directly apply to the computing system without local interpretation. Thus, creating a system-specific security policy is a task that the security manager cannot avoid. (See Essay 5 on multiple levels of policy for additional information.)

Unfortunately (for the security manager), long-term stability is not a property of computer systems. Every system breaks. Every system failure is ultimately repaired by the addition of hardware or software that makes the system different from how it was before the failure. Furthermore, every system is developed and changed over its lifetime. As new functionality is added, the system policy may have to change, and its detailed interpretation will certainly change. Thus, throughout the life of a system, the security manager must continually redevelop and extend the detailed security policy that it implements. Ideas that have been made obsolete by changes in the network have to be changed and redeveloped. The detailed policy also has to be expanded to cover new hardware, new software, new functionality, new connections, and sometimes new kinds of users.

System security evaluation is also part of security management. Evaluation is a continuous process beginning before the system is turned on and ending only after it is turned off to be removed. As development occurs, the policy may change, the protected assets may change, and the threat environment may change. All of these changes alter the security requirements, and they force reevaluation of the system against the new requirements. Even during normal operations in the periods between changes, the security manager must continually reevaluate the system. Unauthorized attempts to access the system also require the security manager to reevaluate the system. The judgment that continued operation will not harm the company is made anew every day. Most of the time, this is an analytic activity, but occasionally some system testing is required.

The scope of these activities is much broader than simply watching over the operation of the system and issuing a few passwords. The security manager must be prepared to be involved in policy development, threat and requirements analysis, configuration and usage management, hardware and software evaluation, and recovery management. All of these areas must be addressed to satisfy the basic job requirements.

### **A few of the facts of life**

The life of a security manager is not as straightforward as deciding what should be done and then demanding that it happen. There are the normal constraints of lack of funds to do what should be done, lack of people to make it happen, and the desire of the users to do almost any-

thing else. There are also constraints that arise from the current realities of computing, and these also have to be taken into account. A few of these constraints are described below.

The catch phrases used as headings are, of course, not literally true. They do, however, describe several facts about the current world that cannot be ignored. They are also suggestive of a useful attitude. The security manager should not become mired in the details of the operation but should always look at the bigger picture.

**Software rots.** A friend uses this phrase to describe the fragility of old software that has been enhanced and repaired many times. After a while the code becomes unmanageable, and new enhancements or repairs tend to result in more problems than they fix. Finally a time is reached when no one will touch the code anymore for fear of breaking it again.

The same phenomenon occurs in systems. It is largely a result of increases in internal complexity that occur when new functions are patched on top of old. As old pieces of the system are asked to do just a little bit more than they were originally designed to do, they necessarily become more complicated and harder to maintain. Over a decade or more this can happen to so many parts of the system that it can no longer be maintained at all.

The security manager cannot prevent this phenomenon from occurring, but strong security management can prolong the life of the system by resisting gratuitous increases in complexity. Systems break. This is usually bad for security. Complex systems break more often and are harder to fix. This is worse for security. Thus, it is in the organization's security interest to keep the system as simple as possible, and doing so is the responsibility of the security manager. Over the long term, complexity management is the security manager's most important function.

**Everything is connected to everything else.** The term "stand-alone computer" is an anachronism that has almost no place in the current environment. The security manager used to be responsible for watching the site's "Central Computing Facility" (CCF). This was all she or he had to worry about, but those days are gone. This is no longer a reasonable approach.

For many years at the Los Alamos National Laboratory, all of the computing capability was located in the CCF. This was true in 1950, and it was true when I arrived in the early 1970s. It changed rapidly in the 1980s. By 1990, when there were about 65 "Cray-1 equivalents" of raw computing power in the CCF, there were over 100 Cray-1 equivalents sitting on people's desks.

In terms of all the measures used by security managers — number of users, number of machines and sites to be protected, volume of audit data, number of network connections — the Los Alamos network ex-

ploded. All of these measures grew by three orders of magnitude between 1975 and 1985. For example, the number of network users grew from four in 1974 to over 4,000 a decade later.

Los Alamos was certainly not alone in experiencing explosive growth. It was a property of the whole world of computing in the 1980s. Explosive growth continues into the 1990s. The Internet had over 500,000 nodes in 1991. The number grew at about 10 percent per month during 1991 and 1992. By the end of 1992, the growth had slowed to only about 1,000 new hosts per day.

But the visible connections are not the only ones that have to be considered. Even if electronic links such as local area networks are absent, almost every small machine has a modem, and sharing of diskettes is pervasive. If data moves between machines, they must be considered connected, no matter how the data gets from one to the other. The components managed by a single security manager are a small portion of a vast network with no easily discernible boundaries. This means that security must be provided without the benefits of isolation.

**Significant computing power is everywhere.** An Intel-486-based machine running at 33 MHz with sufficient memory is roughly equivalent to a Cray-1, the supercomputer of 1980. Even small machines (that is, desktop machines) have enough computing capability to do significant damage to an organization's security. We used to depend for part of our security on the inability of a user to do much computing on his desk. Unfortunately, this simplification no longer works. Today, at some sites the computing power is fairly uniformly distributed, and the machines at the central facility are not much more powerful than those on people's desks.

If computing power is everywhere inside an enterprise, this will surely be true outside as well. This means that the system's adversaries have more capability, and more computationally intensive attacks can be expected in the future. Since it is no longer safe to hide behind complexity, stronger protective mechanisms will be required. Stronger mechanisms will sometimes include better controls within computer products. More often, however, needed isolation between functions — security-critical functions and user processing, for example — will be obtained architecturally.

**Hardware is free.** Some hardware — supercomputers, for example — appears to be very expensive. In an organization with a small budget, even a personal computer may appear expensive. But even the hardware costs of supercomputers are small when compared with the associated costs of buildings, installation, maintenance, software, and people. In a large enterprise, the cost of small machines is in the budget noise. Even in a small enterprise, where the budget arguments are more difficult,

small machines are often on every desk. This means several things for the security manager.

First, since small machines are cheap, the growth of the network will be, to a first approximation, uninhibited by cost considerations. Cost may slow growth but does not prevent it. One should expect continuous growth. As long as the growth doesn't include structural change, however, it presents only mechanical problems (more users to service, more audit trails to process, and so on).

Second, it is easy for internal developers to obtain small machines. Thus, the pace of development is controlled more by the size of the staff than by the cost of hardware. Once again, if the development doesn't include structural change, it presents only minor problems. But if the network wasn't carefully planned in the first place, significant functional change can occur through the addition of small warts. Those development warts are the changes that eventually make the whole system too fragile to be maintained.

Third, it is also relatively easy for security people to get small systems. Often a security-critical process cannot be adequately protected if it has to share a processor with other activities. This is because needed isolation cannot be provided within a single machine. The right solution may be to provide it with its own processor. Often, this is much less expensive and more effective than trying to upgrade the security of the shared processor.

**Developers are everywhere.** The pace of development in the commercial hardware and software world is enormous, especially compared with the pace of security development. There are orders of magnitude more development people than there are security people. And it often seems that everything they do is specifically designed to make the security job harder. Worse yet, detailed security evaluation is expensive and slow. Detailed security evaluation of a new piece of software, when it is possible at all, takes approximately one software generation. That is, the evaluation will be finished at about the same time that the software is obsolete. Hardware is even more difficult and costly.

Change is nearly continuous in any large computing system, and the security officer must continually and rapidly evaluate proposed changes to determine if they should be rejected, be allowed to proceed, or be slowed to allow time for deeper evaluation.

**Love is blind.** When and how to evaluate are difficult problems. Developers have infinite confidence in their abilities and will always argue that evaluation is unnecessary. But, being absorbed in the beauty of their own creations, they will also propose outrageously dangerous ideas and fail to see obvious problems that should be repaired. Developers who normally are only marginally able to communicate in their native lan-

guage can become passionately eloquent if the behavior of their creation is questioned. Evaluation is one area where a security manager should be rigid. All security-relevant code should be evaluated by someone other than the developer — no exceptions.

There are two important aspects to the evaluation problem: the initial baseline evaluation, and evaluation of development and maintenance changes. Both are important and difficult. Unfortunately, they are different, and one set of tools and techniques does not suffice for both problems.

The initial evaluation becomes increasingly difficult as the complexity of the system grows. A large system, such as the Los Alamos network or the network belonging to a large commercial organization, is not amenable to complete evaluation using the tools and techniques available today. It is simply too large and too complex. Even if it could be evaluated, however, it would undergo thousands or tens of thousands of changes during the evaluation period.

The rate of change in the Los Alamos network is tens to hundreds of changes per month. To install changes at this rate without evaluation is unacceptable. To incur the cost and delay of using presently available tools is also unacceptable. Thus, it is necessary to create new tools, speed up existing tools, and make the tools available to a less sophisticated class of users than the people who built them. In the interim, until the needed tools are developed (possibly a long time), it will be necessary for security managers to decide with almost no concrete information which changes have to be evaluated (and how carefully) and which can be installed without evaluation.

New tools are also needed to handle evaluation of maintenance changes. Most maintenance changes are minor and have no effect on system security. However, simply ignoring minor changes can be a big mistake — tiny changes can dramatically affect system security. A way must be found to evaluate the semantic effects of change rather than just the syntactic effects. We must be able to determine the importance of small changes to very large systems quickly and easily.

While we are waiting for better tools, much more effort needs to be spent on informal and semiformal evaluation techniques. Formal evaluation of large connection-rich networks will not be feasible for a long time, but large connection-rich networks are being built and used. Techniques must be available for analyzing them. Most of these networks do not need formal analysis, but they all require some protection and some analysis.

Formal, or even informal but detailed, security evaluation is not practical for any but the smallest or most critical networks today. There is no evidence that this will change anytime soon. A security manager with protection responsibilities today must find some way to do the job without detailed evaluation. Unfortunately, there are no systematic ways to

do this. The result is usually a very conservative attitude about what can be allowed.

## **A strategy**

The picture painted above is pretty bleak. Implementing and maintaining security are difficult and are getting more difficult rapidly. Data in large volumes is easily portable and easily changed, so simple isolation no longer works. The market for security products is still small and cannot keep up with the pace of change.

There is a strategy that will work. Interestingly, it relies on more distribution of computing rather than less. In fact, it requires distribution and decentralization of everything: computing power, security mechanism, and security management responsibility. The elements of the strategy are outlined below. The elements will be weighted differently, depending on the size and complexity of the system. Security managers of small systems may use some of these ideas very little, but even a large and complex system can be protected using this strategy.

**Decentralize.** We are at the end of a decade of massive decentralization of computing power. What used to be concentrated in one place is now spread out over the whole enterprise. Much of the decentralization occurred before we realized that security mechanism has to be distributed along with the computing power. Almost all of it occurred before we began to realize that security responsibility also has to be distributed with the computing power. One manager cannot be directly responsible for the activities of 1,000 workers, and, for the same reasons, one security manager cannot be directly responsible for the security of 1,000 workstations. The responsibility to protect must go with the power to compute.

**Use the system architecture.** Security mechanisms need isolation. If users can manipulate or change the security mechanisms, they lose their value because their properties are unknown. You can't depend on something you don't understand (massive application of faith is not generally considered a valid security strategy).

All security mechanisms are not equal. Some require better protection than others. If isolation is moderately important, improve the security in a shared processor. If isolation is really important, put the security process in its own processor. Remember that hardware is (close to) free.

It is generally true that putting a control process in its own processor improves security. Information hiding is recognized as an important means of managing complexity in the software world. The modern expression of this principle, object-oriented design and programming, is seen as a very exciting way to rapidly produce correct software.



The analog in the network world is to put an object in a processor. The benefits of information hiding accrue at this level of abstraction, just as they do in software applications. The “isolated” process is still subject to all of the tricks and manipulations through its interface that previously worked, but several avenues of attack have been removed. It is much less likely, for example, that the process’s data structures can be manipulated directly than if it shared a processor with a potentially hostile user process.

Of course, in any specific situation, there are subtle manipulations that might be possible that would counter the claims made above. These are, however, second-order effects. Most security managers never have enough time, and little occasion, to consider second-order effects. This, like considering weaknesses in the gate when a fence has yet to be built, is a waste of time.

**Write a threat statement.** Threat can be analyzed in two ways: looking out and looking in. When one looks outward, one gathers information about what potential adversaries might try to do by looking at available intelligence information and at historical information about similar systems. This provides a sketchy picture of a threat environment that may be similar to the actual environment for the system in question. The picture, however, is necessarily incomplete — you cannot know exactly who all your adversaries are or what they might do. (Too bad — if the picture were complete, life would be simple.)

Looking inward, one asks which threats, if they were carried out, would be most damaging. Where possible, it is useful to validate the analysis with available historical or intelligence information. One then proceeds to install protective measures against the most damaging potential threats.

The threat statement should be a specific and detailed statement of what adversaries may try to do and how they might do it. Statements like “Espionage agents will try to obtain sensitive data” or “Criminals will try to steal money” are of no use in selecting controls to counter the threats.

Although it may seem difficult and perhaps unnecessary, the threat analysis must be much more specific. This is important when the implementers are designing specific security features. The threat statement is a road map for system designers. It is important because if you don’t know where you are going, it is hard to tell whether you have arrived. The following can be used as tests to help decide when the threat statement is done:

- A test plan outline should be directly derivable from the threat statement. That is, a reader should be able to see exactly what threats the target is expected to counter and, therefore, should be able to see exactly what tests are needed to verify that the system does what is expected.

- The reader must be able to see the rationale for every required security feature. The reader must understand what threat or threats the feature counters and whether the feature does this alone or together with other features. These linkages should be exhibited explicitly by the authors.
- The described threat must still be clear in 10 years. See the first item above. This is partly a matter of language, partly a matter of presentation.
- Two completeness tests must be satisfied. These two forms of completeness are essential to the process of vetting the system design to determine if it should be implemented.
  - It must be possible to assess the completeness and reasonableness of the threat analysis. The reader must be able to look at the analysis and determine if any important threats have been left out or if threats that have been included are unimportant.
  - The reader must be able to examine the security features that have been specified to determine that all the threats the system is supposed to resist have been addressed adequately.

**Avoid details.** It is not always possible or desirable to avoid all details when managing a security program. Details are important. Yet a security manager is busy, and if all of her time is spent on details, the larger issues will escape attention. A security manager immersed in details has already lost. The following paragraphs give some hints about strategies that can be used to avoid this problem.

*Structure the system as large components.* The properties of a complex system cannot all be understood by a single person. But failure to understand how the system behaves can be fatal. Therefore, the security manager must always simplify. One simplification that helps is to structure the system so that many components contribute to a single kind of processing that can be thought of as a unit. For example, all of the components serving a particular kind of user might be grouped. If a group of components is implemented so that the group has a well-defined interface to the rest of the system, then it can be thought of as a single item — for example, as a single computer. This is just another application of object-oriented thinking. Details are hidden inside the object and can be thought of separately. The decoupling of different parts of the system that occurs when this technique is applied is an enormous simplification. It can be the difference between a system that is secure enough and one that is not.

For example, the Los Alamos network, which contains thousands of computers, supplies only four different kinds of computing (based on data sensitivity and user access). As a result, components can be grouped into a handful of computational structures, each of which can be thought of as an entity. Relative freedom of communication and functional development is allowed within an entity. But strong justification and detailed scrutiny are required for any new connections proposed between entities.

*Use triage.* “Every change to a system should be analyzed to determine that it has no adverse effect on security.” This is a great idea, but it is impractical for any system larger than a few PCs. For a large system, it is impossible. Still, the security manager cannot delegate too much authority to the developers. I once discovered, at the last minute, that a critical security process was about to be changed without any independent testing. Even though the process had been reimplemented in a new language, and it was to run on a new machine with a different operating system, the developers didn’t consider the change to be security relevant because no functionality was supposed to change.

The security manager should be aware of every change to the system but must learn to sort them quickly. Three categories are used: changes that are unlikely to affect security and should proceed without interference, those that have security implications and should be evaluated, and those that are outrageous and should be rejected without study. Although this takes experience, it becomes easier with time and is surprisingly effective.

*Use the environment.* Every system is different. Its purpose is different. Its users are different. And, its environment is different. The environment includes all of the context in which the system operates and must always be considered when the threat statement is written. No system can provide all of the controls that are necessary to protect the information processed in it. Some controls must always be placed in the environment.

Since the environment must contain some of the controls used to protect data, it is reasonable to consider using the environment whenever new or additional controls are needed. Often it will be cheaper and more effective to implement new controls in the environment instead of in the system. The point is that this option should always be considered. It is not necessary to tweak the system every time protection has to be improved.

**Use deterrence.** Security people like prevention. If something is not supposed to happen, don’t let it happen. But prevention isn’t always possible. Some undesirable actions are a result of an authorized person doing more than he or she was authorized to do, or doing something for the

wrong reason. The computer system cannot be expected to tell the difference. The user can do what the user can do, and motivation cannot be assessed.

If prevention isn't possible, try for detection. Audit trails can be created for most actions and later processed by machines or by humans to detect actions that a user should not have taken. But detection sometimes isn't possible either. This might happen because the relevant event can't be captured, because it gets lost in a mass of detail, or because the volume of related events precludes inspection of each one. For example, the file storage system at Los Alamos holds several million files. While it is likely that a few of these contain unauthorized data, periodic inspection of them is out of the question. In a case like this, deterrence is the only solution. Warning notices, spot checks, and public hangings all serve to remind users that there are rules of conduct and that someone cares how they perform.

**Guard the borders.** The system being protected is probably connected to other systems, all outside the control of the local security manager. Data may flow regularly to and from outside systems in the performance of the company's business. The situation can easily become amorphous and highly fluid.

It is important for the security manager to establish a security perimeter outside of which control is not exercised. It is, of course, important to protect the perimeter well enough that control is not necessary on the outside. Three tools can be used to help accomplish this:

- Use strong authentication procedures for traffic crossing the perimeter. Authenticate everything if that is possible, even system-to-system transfers. Where the external connection is to the Internet, it is important to consider carefully which protocols should be passed through the perimeter, which should be buffered, and which should be stopped.
- Buffer critical decisions and critical transfers. Don't implement critical decisions about data movement across the perimeter in the processors that are on the perimeter. Instead, make them further inside the perimeter and validate their correctness redundantly in the processors on the perimeter.
- Audit more completely at the perimeter, and make it clear that auditing takes place. This will help to identify when something has failed. It will also help with the subsequent damage assessment, and it will serve as a deterrent to malicious activities in the first place.

**Evaluate.** I pointed out earlier that complete evaluation of complex systems is impossible. This means that it would not be possible to get the system to hold still long enough to apply any systematic technique to it in depth. It does not mean that evaluation cannot or should not be performed. In fact, evaluation must become a continuous activity of the security manager. The current state of the system is examined daily (a basic security responsibility), and changes are made as necessary.

The questions to be asked are:

- Has the threat environment changed in a way that calls for reassessment of the security controls in place?
- Has the system changed in a way that might create new vulnerabilities or exacerbate old ones?
- Has the erosion of security due to accumulated changes moved the system outside the parameters set by management?

If any of these questions is answered “Yes,” action is required. The action could range from informing management to shutting down part or all of the system, but action cannot be avoided.

## **Summary**

Computer users have gotten connected. They have computational capabilities on their desks that are electronically linked to everyone else’s capabilities. This happened because the vast amounts of information available after connecting and the gains in productivity make the risks worth taking. A “trivial” consideration like security will not reverse, stop, or even slow the trend toward even greater connection.

Security must be provided even when complete isolation is no longer possible. This is difficult, but not impossible. Unfortunately, the security techniques of the 1970s and the early 1980s are either no longer applicable or no longer work. Isolation and centralization of processing no longer apply. Reliance on ever-stronger controls built into general-purpose operating systems no longer works because a large number of connected systems makes it impossible to understand how they should relate to each other.

Instead we can turn to object-oriented techniques. We encapsulate important processes in their own processors to create local isolation and take advantage of the benefits of information hiding. We build larger objects out of smaller objects and control the ways in which they can interact. And, finally, we control and monitor the interactions between the larger objects and the outside world.