Cybersecurity Considerations for Information Systems

Cynthia E. Irvine
Center for Information Systems Security Studies and Research
Department of Computer Science, Naval Postgraduate School, Monterey, California 93943

Abstract

The significant efficiencies possible through the use of information technology in public systems are alluring, however, as the value of the information stored electronically increases, computer systems become targets for abuse and attack. To ensure continued public confidence in these systems, managers need to understand the impact of security shortcomings in their automated systems. A high level taxonomy of threats to information systems is presented to provide a basis for security requirements. Fundamental concepts of computer security are reviewed. The costs and benefits of investment in cybersecurity will be introduced. The concept of organizational information policy, mechanisms for its enforcement, and the value of assurance and the notion of costs and benefits of investment in cybersecurity are presented.

I. Introduction

Advances in information technology (IT) have resulted in a transformation of business processes. With automated support, it is possible to rapidly adapt production and distribution to changes in market demands, with resulting increases in enterprise productivity. Although organizations have embraced information technology and have integrated it into their information management activities, they have done so with caution. For example, the Internet may be used to supplement public relations and marketing, and to provide an interface for customers or clients, but core business activities are isolated from and do not rely upon the open Internet. Instead, organizations achieve widespread or global connectivity by combining corporate local area networks (LANs) with leased wide area systems or virtual private networking solutions. These network architectures are complicated and usually expensive. Organizational networks, although key to the growth and effectiveness of many businesses, have yet to fully utilize the Internet, and, for many enterprises, the promise of the Internet remains a chimera. This situation has arisen because a key enabling technology has not been adequately addressed. That technology is security.

Constructing systems with more than superficial assurance that security will be correctly enforced is difficult. Historically, the desire for rapid advances in functionality and performance have resulted in little attention to security on the part of commodity IT product vendors. Many fundamental security concepts have been known for decades, yet the extra effort required to construct secure systems has precluded their development for environments where security did not "appear" to be critical. Unfortunately, we have arrived at a point where lack of security is overwhelming many organizations. Spam is becoming an increasing drag on worker productivity: employees are bombarded with distracting missives and must sort through dozens of irrelevant messages to find work-related email. Rampant and rapidly propagating worms and viruses can cause unpredictable discontinuities in business processes at critical junctures. Popular

operating systems, applications, and their respective updates are of low integrity: opportunities for the insertion of malicious code into the heart of one's systems abound.

Organizations are increasingly moving toward the use of computers as not only repositories for public records but as a distributed interface for update and maintenance. For example, in California, county recorders are actively pursing electronic recordation of real estate transactions (Bernhard 1999). Clearly, this results in significant cost savings in terms of handling and paperwork, yet such systems, despite claims that the use of cryptography makes them "secure", create new opportunities for fraud and malice, the consequences of which, especially in the case of real estate transactions, are dire and difficult to undo. It is imperative that those responsible for the public trust understand the repercussions of these changes. Even though records might be maintained in a relatively secure repository, a questionable assumption at best considering the numerous flaws found in typical commercial systems, the many systems that might provide input to these records and the networks that connect them are patently insecure.

The vulnerabilities of current IT systems have become regular fare in the press. Hardly a day passes without several news items or articles about some security bug or exploit. So managers have been forced into a Procrustean bed: they must use IT to be competitive but in doing so appear to be doomed to languish in an unending purgatory of system attack and repair. To meet due diligence requirements, corporate information officers in all sectors must take measures to protect their networks and to create better security architectures. With no assurance regarding the security qualities or even the provenance of software and systems, system owners have few components from which to construct sound security architectures. Consequently, we have entered a period of cybersecurity uncertainty.

Today, software and systems are created with disclaimers telling the public to use these technologies at their own risk. This state cannot exist indefinitely, and litigation is inevitable. Within cyberspace disputes will arise regarding the ownership of information, protection of information from theft and trespass, and liability from system or software malfunction. Security measures will be subject to legal scrutiny. For computer systems to be operated within a litigious context, organizations will be required to articulate corporate information policies so that computer system implementations are consistent with organizational rules and directives. In addition, accountability mechanisms for automated systems will be required so that disputes can be resolved. Computer security involves the design, implementation, and operation of computers to permit only authorized disclosure or modification of information. Often included under the rubric of computer security is the subjective notion of availability, i.e., providing some guarantee that when computing resources are needed they will be present. Gligor (1983) argues that there is no mechanism or model that is able to provide necessary and sufficient solutions to this problem. Computer owners should keep this in mind when making legally binding promises regarding availability.

It is also important to carefully distinguish between software safety (Levenson 1995) and computer security. Safety provides assurance that a computer program does what its owners intend it to do. Security requires that systems exhibit no unintended functionality. In a sense, the problem of security is to demonstrate that the system does not do (or permit) anything that is not in the specification. It is a negative requirement. Of course, since we would like the security mechanisms to work correctly, computer security engineers use many of the concepts of safety and software engineering to design and construct secure systems (Denning 1982, Gasser 1988; Pfleeger 1996; Summers 1997).

Computer and network security can be divided into three synergistic areas: security within computers, security for communications between computers, and secure management of systems. These elements overlap and must be in place in order to achieve a secure operational system (National Computer Security Center 1994). To assess the security of an IT system all elements must be reviewed. The weakest component of the overall security posture is likely to be the avenue for attack. This essay will focus on technical aspects of security in computers and will touch upon emerging concerns and costs associated with computer security. First, we must understand the computer security problem.

II. Computer Misuse: Threats and Vulnerabilities

Misuse of computers can include theft of machine cycles, unauthorized use of peripherals, modification of critical data, or capture of sensitive or proprietary information. Computer misuse results from the existence of one or more system vulnerabilities that can be exploited by a threat, where a threat is an active entity that exploits a vulnerability to gain unauthorized access to system resources.

Brinkley and Schell (1995a) describe several forms of computer abuse. All result in the unauthorized modification or disclosure of information and are briefly presented here.

Human Error. Mistakes in system usage are unavoidable but may open a window of opportunity allowing an attacker to exploit an error-caused vulnerability. The attacker will have to wait until the vulnerability appears, so the probabilistic nature of vulnerabilities resulting from human error may render them less attractive than more deterministic approaches. A combination of user training, appropriate user interfaces, and security controls can help minimize user errors.

User Abuse of Authority. Here an authorized user misuses permissions, "cooking" financial records or other information. Intricate schemes may be developed involving several accomplices. Security controls can restrict privileges and provide an automated audit of the activities of authorized users.

There is, an even darker side to insider activities (Winkler 1997; Denning 1999). Sensitive or proprietary information is highly vulnerable to individuals who have been compromised by an adversary. Treason, whether to the state or a business, can be deterred by background checks and various technical mechanisms and procedures; however, if the traitor both has access to a particular piece of sensitive information and is willing to steal it at any cost--an information kamikaze--then, although subsequent capture may be guaranteed, complete protection against compromise cannot be provided: an insider can memorize the information and sell it on a street corner.

Direct Probing. This form of abuse involves the manipulation of permitted functions presented at the system interface in unintended ways. Stories of careless or inadequate system administration abound. Systems installed with default or no passwords are unfortunately common. The assumption that the computer systems will be operated in a benign environment renders them vulnerable to probing attacks, sometimes from across distant networks (Stoll 1989, Denning 1990). Often one or more hackers issue inappropriate commands to the target system in order to gain control over selected system assets. Both careless administration and the inability to impose fine-grained security controls, i.e., management of permissions on a per user basis rather than on broad groupings of users, create an environment that invites direct probing.

Probing with Malicious Software. As in the previous case, the computer is probed in ways that are permitted, but not intended. However, in this instance a well-intentioned user of the target system becomes an unknowing accomplice. This is accomplished using Trojan Horse software, i.e. software that provides some "normal" function that serves as a cover for its clandestine, malicious function. The unsuspecting user executes the "normal" function and, in the background, the clandestine function executes with the user's current permissions, all within the context of the user's authorized activities. Such software can consume machine resources, copy, or modify sensitive information. If auditing is turned on, the victim may be blamed for espionage or sabotage, yet the victim had no idea that the Trojan Horse was hidden within the "normal" software. Clearly, issues of accountability are muddied when Trojan Horses can execute behind the scenes.

There are many breeds of Trojan Horses: viruses, worms, and logic bombs. Hoffman (1990) and Denning (1990) provide many examples of malicious software that are Trojan Horses. The increased use of downloadable software and mobile code without commensurate mechanisms to assure its origin or integrity pose serious threats to typical user systems.

Direct Penetration. This form of misuse involves the exploitation of a flaw in the system's security mechanisms. Here administration of the system may be perfect, but an error in design or implementation makes the system vulnerable to attack through exploitation of that flaw. Starting in the 1960s tiger teams conducted successful penetrations by exploiting software and hardware flaws (Anderson 1972). System programmers would patch the flaws, but these activities invariably resulted in the creation of new flaws that were subsequently attacked. The result was a game of "penetrate and patch", always won by the penetrators who only had to find one exploitable flaw, and lost by system owners who needed to find and repair all flaws. The Multics vulnerability analysis (Karger and Schell 1974, Karger and Schell 2002) and Landwehr et al.'s (1994) software flaw taxonomy provide background material illustrating the problem.

Subversion of Security Mechanism. This is the insertion by an adversary of a *trapdoor* within the system that will provide a toehold for subsequent system exploitation. In a now classic analysis, Myers (1980) describes the many ways that a system can be subverted throughout its lifecycle. The ease with which a highly effective artifice, undetectable through system usage monitoring can be inserted into a system has been demonstrated (Anderson 2002, Lack 2003, Murray 2003, Rogers 2003).

Once an artifice is inserted, it can be virtually impossible for system security personnel (or even system designers) to locate. A common artifice is a trapdoor that permits an adversary undetected access to the computer system. Karger and Schell (1974) suggested a particularly insidious trap door, which was made famous by Thompson (1984). The trap door replicated itself, but could only be found by inspecting the executable code for the Unix operating system and its compiler. Other artifices may activate as a result of triggering events to cause system malfunction, information corruption or exfiltration. Modern commercial operating systems consist of tens of millions of lines of code and provide ample hiding places for subversive code. In fact, many vendors willingly permit their system developers to put artifices into commercial products. Many operating systems and applications are already known to contain large unexpected code modules that have been nicknamed "Easter Eggs" (Wolf 2003).

III. An Alternative

The examples of computer misuse and the experience of the penetrators and tiger teams leads us to conclude that building a flawed system with the hope of later securing it will fail. Instead, a constructive approach to secure system design and development is recommended. A notion that has been subjected to scrutiny and continues to be found valid is that of a security mechanism that enforces a policy relating to the access of active system entities, which we can loosely think of as processes or programs in execution, to passive system resources (objects) based upon authorizations. The Reference Monitor Concept (Anderson 1972) is an abstract notion that articulates the ideal characteristics of a reference validation mechanism:

- It is tamperproof.
- It is always invoked to check access requests.
- It is small enough to be subjected to analysis and tests the completeness of which can be assured, thus providing confidence in its correct enforcement of policy.

The objective of the secure system designer is to use the Reference Monitor Concept as an ideal to which actual system implementations strive. The degree of success in realizing the objectives of the Reference Monitor Concept becomes a measure of confidence in a systems ability to enforce organizational security policy.

There are several stages in the development of a secure system. First, the organization's security policy for people's access to information must be articulated and translated to a computing context. Second, a system must be constructed to enforce the security policy. To address the threats of direct penetration and subversion, system builders will need to provide some level of confidence that security mechanisms actually enforce the policy and that the system developers have limited the possibility of flaws and artifices. This construction process is analogous to those students create in geometry. Their proofs are by a constructive demonstration rather than by induction. To demonstrate that a system is secure, it is necessary to demonstrate that all of the code in the system is supposed to be there and is needed to do the job, and that there is nothing extra. Extra code equates to unspecified functionality, which might include trapdoors or intentional flaws built into the system. Third, the system must be maintained and managed so that the possibility of accidents or subversion is minimized and personnel understand how to use the system properly. Finally, the system should be subjected to third-party review. For many, computer systems are black boxes. We cannot examine proprietary source code to ensure that the vendor has built the system as claimed. Fortunately, an evaluation process is in place that gives those acquiring systems a measure of its security functionality and assurance.

IV. Security Policy

Articulation of a security policy reflecting the way an organization actually treats its information is essential to computer security. Only when the policy has been clearly expressed can systems be designed and implemented to enforce it.

Understanding organizational policy is not easy. Sometimes it is unwritten, but often the documented policy lies at the bottom of a desk drawer untouched since its formulation. Policies should be stable (policies that permit frequent changes in permissions to access to information will be described later), but they may gradually evolve. For example, when a startup business is small, access to personnel and accounting records may be authorized for a single administrator. Two years later, when the business has hundreds of employees, different departments may

manage personnel and accounting and access restrictions between departments are probably appropriate.

Stern (1991) notes that organizations must state security policy objectives, an organizational security policy, and an automated security policy. These move from high-level statements to specific rules and restrictions pertaining to automated systems and are reviewed below.

A few of the more obvious questions to be answered when attempting to articulate security policy include: What information is to be protected? Who is to be allowed access to a particular item or set of information and what will the user be permitted to do to it? What rules or regulations will be invoked to decide who has access to specific information? Does the policy that has been described on paper actually reflect what the organization is really doing (or should be doing)? If an organization does not engage in secure information management in the physical world, then it is unlikely that a translation of these practices to cyberspace will result in any improvement. Sometimes management will state requirements for its computer systems that are not part of the organization's business practice. For example, suppose high-level policy states that no information is to be exchanged between the human resources and the public affairs departments. Also suppose that actual practice involves regular movement of information between the two departments. Then a new computer security policy prohibiting this flow of information across the network will either be ignored and useless, or it will render daily operations cumbersome and infeasible. A review of security policy may offer an opportunity to effect improvements in the way business is conducted, but the computer should not be viewed as a panacea for ingrained, sloppy information management practice.

There are two fundamentally different types of security policies and the mechanisms for their enforcement differ: non-discretionary and discretionary.

Non-discretionary security policies are global and persistent policies relating the access by individuals to information. Information is organized into broad equivalence classes, such as SECRET and UNCLASSIFIED or PROPRIETARY and PUBLIC. The intent of mandatory policies is to describe the permitted information flows between these classes.

As an example, consider an organization that has partitioned its information into two integrity-related closed-user-groups: Trustworthy and Pond-Scum. The integrity attributes of information are maintained throughout the organization and do not vary by time of day, or by the day of the week. In the paper world, specific users are given authorization to view and modify information. Only those who are highly trusted are able to change Trustworthy documents. Since these trusted users exercise judgment, they can be relied upon not to enter information of Pond-Scum integrity into Trustworthy documents. Similarly, within a computer a trustworthy authorization level can be assigned to an active entity permitting it access to certain trustworthy documents. Unfortunately, software does not possess judgment and could even contain a malicious Trojan Horse intent upon corruption of Trustworthy documents. Thus, in a technical expression of the policy, trustworthy processes are not permitted to read Pond-Scum (Biba 1977). The Trustworthy processes can, of course, read information of even higher integrity and they can "improve" Pond-Scum by writing Trustworthy information into the Pond-Scum domain. In addition Pond-Scum entities can always improve themselves by reading from Trustworthy objects and modifying Pond-Scum information accordingly. Our rules will also confine trustworthy processes to read and execute only trustworthy code. So if code certification and authentication methods were available, trustworthy code could be identified.

Similar policies can be constructed for confidentiality. The most familiar is the military classification system with TOP SECRET, SECRET, CONFIDENTIAL, and UNCLASSIFIED labels. Equally useful labels can be applied in the commercial setting by changing the names to PROPRIETARY, COMPANY SENSITIVE, and PUBLIC (Lipner 1982). For confidentiality, the read/write situation is turned upside-down (Bell and LaPadula 1973). Entities having low secrecy labels cannot read information with high secrecy labels, but high secrecy entities can read low secrecy information. Trojan Horses are thwarted by barring active entities at high secrecy levels from writing to objects at lower secrecy levels; however, it is permissible for low secrecy entities to write to high secrecy objects. In all cases, both active and passive entities are assigned immutable labels that are compared when accesses are attempted.

A mandatory security policy can often be recognized by the degree of damage its violation would cause the organization and the punishments associated with policy violation. Policy violations that would result in the financial ruin of the organization and imprisonment or dismissal of personnel are likely to be mandatory. Less severe punishments are meted out for violations of the second general type of policies: discretionary policies.

Discretionary policies are neither global nor persistent. Access to information can be granted or denied by an individual or a process acting on behalf of an individual. This ability to change access rights “on the fly” makes discretionary mechanisms very flexible, but also vulnerable to attacks by Trojan Horses. Malicious software can act in a manner dramatically opposed to the user's intent.

A real example illustrates this point. Today web sites provide active code modules, or applets, that are automatically downloaded to the user’s web browser and executed by browser-enabled software packages. A group of hackers constructed a web site that contained executable content. When victims accessed the site, not only did the executables provide them with the expected service, but, behind the scenes, a Trojan Horse with the potential to change permissions installed an artifice in each victim’s financial package. When the financial system was activated, funds were transferred from the victims’ bank accounts to that of the hackers. Although the victims may have had discretionary controls on their financial programs and files, the Trojan Horse was executing with the victim's authorizations. So, the Trojan Horse was able to manipulate the access controls and insert its artifice. This example also illustrates the powerful benefits of non-discretionary controls. Had mandatory integrity controls been in place, the malicious software (Pond-Scum) could have been contained. With the victim's active entity for web browsing running with Pond-Scum integrity, the Trojan Horse would not have be able to write to Trustworthy.

An examination of organizational risk with respect to policy enforcement in an information system will permit management to understand how measures can be taken to improve the overall security posture. Jelen (1998) provides insights into understanding risk and assurance. Of course, mechanisms must be accompanied by management strategies to insure that external aspects of security complement those within the system.

Once the security policy has been articulated by management, a technical interpretation of policy must be formulated. Technical policies relate the access of active entities to passive containers of information such as files, directories, programs, pages, records, etc. When access to information is requested, there are only two fundamental access requests: read and write. There is no notion of maybe in a computer: a yes or no answer must be returned--either access is granted or denied.

Equivocation can appear to be provided by elaborate software systems built using read and write controls, but fundamentally, the choices are zero or one, read or write, yes or no.

Careful analysis of security policy expressed in terms of a mathematical model can be beneficial. First if the policy is inconsistent, the mathematical proof of the policy will fail. Thus the expense of building a flawed system can be avoided. Second, the mathematical expression of policy provides system developers with a starting point for mapping system design and implementation to policy. This mapping provides assurance that the organization's policy has been correctly implemented.

V. Secure System Construction

Once a formal or informal model has been developed, system designers apply an arsenal of software engineering techniques to construct the hardware and software mechanisms that provide trustworthy security policy enforcement (Gasser 1988; Brinkley and Schell 1995b). In their seminal paper on system security, Saltzer and Schroeder (1975) identified design principles for the construction of secure systems. We will start with these.

Psychological acceptability: The human interface provided by the system should be easy for users to understand and use. Complex interfaces that do not match the user's mental image of protection objectives are likely to be unsuccessful. Instead, protection should seem natural, simple, and efficient, otherwise users will bypass them or avoid the system altogether.

A simple example of a psychological acceptability problem is that of user passwords. A recommended password is eight characters long and consists of upper and lower case characters, numbers, and punctuation characters. If each user is required to memorize several such passwords, it is likely that the passwords will be manually recorded somewhere in the individual's workspace. Thus corporate systems are vulnerable to break-in by anyone with access to the office: repairmen, janitors, etc. A single sign-on mechanism for users would be preferable.

Another area where psychological acceptability plays a role is that of interfaces for discretionary access control management. Current interfaces are difficult to manage and understand. Easy to use interfaces are needed to encourage appropriate use of discretionary controls.

Least Privilege: No individual should be provided with greater access to information than needed to do his or her job. With access limited to the information domain of the job at hand, the potential for widespread damage is limited. Auditing of activities within a particular domain permits system owners to narrow the scope of potential misuse.

Least privilege is essential for circumscribing the effects of malicious software and Trojan Horses. In our web site example, if the system allows users to enter a circumscribed Pond-Scum domain prior to downloading and executing Pond-Scum material, then the effects of the malicious software can be limited and trustworthy domains can be protected from corruption. When applied to confidentiality, least privilege can prevent malicious software from reaching into other domains to grab information for exfiltration to hostile entities.

Systems can be internally structured to enforce least privilege. Such mechanisms can be used, for example, to prevent applications from corrupting libraries and the operating system.

Fail-safe Defaults: Two approaches to granting permission to information are possible taken. In the first case, everyone has access unless explicitly denied. The second, and more conservative, approach assumes that initially all access is denied and that permission is subsequently granted.

Lunt (1989) provides a detailed discussion of these concepts as they apply to discretionary security policies.

The notion of fail-safe defaults applies not only to files and databases, but to entire systems. For example, the concept of trusted recovery implies that, should a system crash, then, when operation is restored, security controls are in place and unauthorized users do not suddenly have unexpected access to system resources.

Complete Mediation: Authorization must be checked for every access to every information object. This notion clearly parallels the reference monitor concept. In a networked system, it implies consideration of a system-wide view of authorization, delegation, and access control. From the enterprise perspective, this view permits a chain of accountability to be established. It also requires careful examination of users' authorizations during the course of a task. If authorizations may change, then access rights should be changed as well. For example, if a user is authorized to access trustworthy information and then wishes to change to the Pond-Scum domain, then prior to changing user authorizations, all write access to trustworthy information should be terminated.

Separation of Privilege: A protocol requiring satisfaction of two or more conditions prior to certain actions provides a more robust security mechanism. For example, a rocket launching system may require two or more persons to agree to its startup. Banking systems may require authorization from both a teller and a supervisor prior to the execution of transactions over a certain amount.

Economy of Mechanism: This principle is associated with the confidence one may have that a protection mechanism correctly reflects policy in both its design and implementation. If the security mechanism is large and spread throughout chunks of a huge system the majority of which provides business functionality and has no relevance to protection, then it will be difficult, if not impossible to locate all of the security relevant components. This will make analysis of the security mechanism problematic. In addition, side-effects from non-security relevant components could render the protection mechanism ineffective. Traditionally, it has been argued that security mechanisms should be implemented in small, complete components at the lowest levels of system hardware and software. Any components upon which the security mechanism depends must have assurance of correctness and tamper resistance at least commensurate with that of the security mechanism itself. Today, the complex mechanisms of application- and sockets-based security share all of the vulnerabilities of the underlying system.

Least Common Mechanism: Physical and logical separation of mechanisms provide isolation of users and their domains from those of others. Variables and resources shared among users can become vehicles for information channels, including covert channels.

Covert channels (Lampson 1973) involve the manipulation of system resources in a manner unintended by system designers to signal information from a sensitive domain to a less sensitive one. For example, a process running at PROPRIETARY could consume all available disk space so that when a PUBLIC process attempts to write to disk it will receive an error message. By consuming and releasing disk space, the PROPRIETARY process can cause the PUBLIC process to receive a sequence of success or error messages when it attempts to allocate disk space. This sequence of messages can be used as a signal composed of zeros and ones to smuggle information from PROPRIETARY to PUBLIC. Thus highly sensitive information such as encryption keys could be leaked. Considerable effort to eliminate or limit the bandwidth of

covert channels is part of the development process of high assurance computer systems; however, there are open research issues in the area of covert channels, so they remain an area of concern for systems containing very sensitive information.

Least common mechanism can also be applied in the context of granting privileges. Mechanisms that permit users to assume privileged status in order to accomplish an ordinary function are, from a security perspective, inferior to those that restrict privileges. The Unix operating system (Ritchie and Thompson 1974) contains the infamous *setuid* mechanism that permits user's privileges to be amplified to those of the system administrator for certain functions (Levin et al. 1989). Flaws in the design and implementation of such privileged modules allow users to break out of the program and have unrestricted access to system resources.

Open Design: If the security of a system depends upon the secrecy of its design, this will be an invitation for espionage on the part of adversaries. By submitting the security mechanism to scrutiny by unbiased third parties, its effectiveness can be assessed and potential flaws identified in advance.

An example from history shows that a closed design is no guarantee of security. Prior to 1949 (Shannon 1949), cryptographic systems were based upon complex, but secret algorithms. During World War II, careful analysis by the Allies combined with a failure on the part of the Germans to recognize weaknesses in their secret cryptographic mechanisms resulted in a decisive advantage for the Allied forces (Kahn 1996). Information theory (Shannon 1948) provides a basis for analyzing cryptographic algorithms and, today, most are available for open review. Today, the secrecy of communications depends upon a mathematical understanding of the strength of the algorithms used, protection of the cryptographic keys, and systems designed so that invocation of cryptographic mechanisms will not be bypassed or subverted.

Organizations obtaining security equipment and mechanisms need to validate vendors' claims. Independent evaluations by unbiased parties are needed. In the absence of independent evaluations, a vendor can claim that a security mechanism solves all problems, while, in fact, the mechanism may do little or might even diminish an organization's system security. In addition, a rating scale is needed so that vendors know when they have done enough to satisfy a specific set of standardized criteria. Using these, the functional mechanisms for security policy enforcement and the constructive security techniques applied during the system design and development can be assessed. The latter provide confidence that the mechanisms meet security objectives and embody the philosophy of the Reference Monitor Concept.

It would seem that open design is an anathema for the vendors of proprietary systems. Fortunately there is an alternative to making all proprietary systems open source. A framework for the evaluation of proprietary systems has been developed that allows security assessments to be conducted at designated laboratories. The Common Criteria (National Institute of Standards and Technology 1999) provide a framework for standards against which secure systems and subsystems can be constructed and evaluated.

Although not principles for secure system design, Saltzer and Schroeder described two notions which merit discussion.

The first is *work factor*. This is often applied in cryptography, where it might measure the number of machine cycles necessary to mount a brute force attack against a cryptosystem. The work factor may cause an adversary to seek a more subtle, but ultimately faster, cryptanalytic

attack on the system through the science of cryptanalysis (Schneier 1996). A secure system's ability to control users' access to information and maintain logs for accountability is not based upon secret keys, but rather upon sound engineering principles used to build a trustworthy mechanism enforcing security policy. A well-constructed system may force the adversary to choose alternative, less costly attacks such as social engineering.

The second is *compromise recording*. This is related to auditing, which records or detects when information has been compromised, but fails to prevent it. Bonyun (1981) discusses issues related to the creation of a coherent, traditional logging process where selected events are recorded for subsequent analysis. In the last section of his paper, he introduces the notion of active auditing, an idea that has evolved into the many intrusion detection systems currently available today. Porras (1992), Lunt (1993) and Amoroso (1998) provide useful surveys of intrusion detection techniques.

A mechanism for compromise recording will depend upon a system that provides protection for its essential elements, e.g. the audit log and the logging mechanisms. An accountability mechanism that lacks integrity may be worse than none. Compromise recording mechanisms must have sufficient assurance to be admissible as evidence in litigation. As an example, suppose that we have an authenticator mechanism (digital signatures and or cryptographic checksums) to set an alarm when information is modified without authorization. Recalculation of the signature or checksum is required to detect the change. Here we must have assurance of the correctness and penetration resistance of the mechanisms used to protect the checksum computations, the cryptographic keys, the rule-based system to flag any modifications, and the logs used to store flagged events.

Authentication is a fundamental requirement for secure systems. There must be mechanisms in place to permit users to be authenticated to systems so that a binding can be established between a user's identity and the processes established to act on behalf of that user. This notion extends from the desktop to those in client-server relationships that may be established during the user's session. To avoid spoofing by malicious software, users need a "trusted path" to their systems so that the systems can be authenticated to the user. Again, this notion extends to client-server relationships. The trusted path may also be invoked during security-critical activities for which clients and servers need to reauthenticate themselves. Reauthentication across network connections remains an area for research and development.

Creation of network connections should be based upon some mutual understanding of the trustworthiness of the remote system. In a heterogeneous environment, some standard attributes that can be used to characterize end-system security properties are needed. Frameworks for trust relationships (Blaze et al. 1996) are still emerging and will require an infrastructure of registration and verification authorities for system identifications and trust attributes. The evolving public key infrastructure is likely to provide a context for these frameworks. Smith (1997) Stallings (1998) and Kaufman et al. (1995) give useful surveys of cryptographic techniques and protocols for networked and internetworked systems. Vehicles to insure that cryptographic keys are recoverable for both operational and legal purposes, while insure privacy for individuals and organizations, are needed.

VI. Cost of Security

Management must be sensitive to the total cost of ownership associated with security mechanisms and procedures. Because many organizations do not want to reduce user confidence

by openly discussing security problems, there are few case studies and the true cost of system insecurity and the cost of installing and maintaining security features are difficult to determine. Acquisition and installation is only a portion of the cost of a security measure. Other costs include:

User Training. If the mechanism is not transparent to users, they must be trained to use it and management must make periodic checks to insure that users have integrated the mechanism into their routine activities. Part of training will include helping users take responsibility for the security of their systems and information.

Decreased System Performance. Many add-on security mechanisms, such as host-based intrusion detection, take machine cycles from productive activities. These must be considered and sufficient resources acquired to support both add-on security mechanisms and processing required for organizational business.

Security Administration. System administrators must be trained to configure and manage security mechanisms. Depending upon the security mechanisms chosen, network connections must be scanned, password systems maintained, encryption keys and devices managed, firewall configurations maintained, virus scanners updated, flawed systems patched, etc. The administration of security controls can be quite time consuming and, if the computer system is a large network, additional personnel may be required. When the network is under attack, security administrators can be overwhelmed.

Audit and Intrusion Detection Administration. If audit logging or intrusion detection systems are deployed, these will require configuration and maintenance. Rules and triggers will have to be set as the security posture of the organization evolves. Audit logs must be reduced and analyzed. Special skills are required to support these often time consuming activities.

Consultants. Where in-house expertise is insufficient, consultants may be needed to conduct risk assessments, devise network security architectures, and assist in handling security incidents or attacks. Both government-supported and private Computer Emergency Response Teams are available to help with incident handling.

VII. Conclusion

A million happy users may be oblivious to exploitable flaws in their systems. The absence of a discovered security bug does not mean that a system is secure. Despite our understanding of the principles of design and implementation of secure systems, commodity products provide mostly superficial security. Thus, system designers often resort to ad hoc solutions. These are intended to lower security risks by creating layers of protection. Unfortunately, if each of these layers it is not truly trustworthy, then it is not too difficult for the layers to be penetrated and the systems overthrown. Managers need confidence that their security policies will be enforced, but they must play a role by carefully describing the policy and insuring that, once systems are in place, the organization embraces the day-to-day challenges of administering and maintaining their secure systems.

References

1. Amoroso, E. G. (1998). *Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Traps, Trace Back and Response.* Intrusion.Net Books.

2. Anderson, E. A. (2002). *A Demonstration of the Subversion Threat: Facing a Critical Responsibility in the Defense of Cyberspace*, Masters Thesis, Naval Postgraduate School, Monterey, California, March.

3. Anderson, JP. (1972). Computer Security Technology Planning Study. ESD-TR-73-51, Air Force Electronic Systems Division, Hanscom AFB, Bedford, MA. (Also available as Vol. I,DITCAD-758206. Vol. II, DITCAD-772806)

4. Bell, D.E., LaPadula, L. (1973). Secure Computer Systems: Mathematical Foundations and Model. M74-244, MITRE Corp., Bedford, MA.

5. Bernhard, T. (1999). An Electrifying Image, California County, California State Association of Counties, Sacramento, CA. Jan/Feb. Also available at http://www.csac.counties.org/counties_close_up/issues_and_trends/electronic_recordation.html (accessed December 2003)

6. Biba, K. J. (1977). Integrity Considerations for Secure Computer Systems. ESD-TR-76-372, MITRE Corp., Bedford, MA.

7. Blaze, M., Feigenbaum, J., Lacy, J. (1996) Decentralized Trust Management, Proceedings of the 1996 IEEE Symposium on Security and Privacy, Oakland, CA, May, pp. 164--173.

8. Bonyun, D. (1981). The Role of a Well Defined Auditing Process in the Enforcement of Privacy Policy and Data Security. Proceedings of the 1981 IEEE Symposium on Security and Privacy, Oakland, CA, April, pp. 19--25.

9. Brinkley, D. L., Schell, R. R. (1995a).What is There to Worry About? An Introduction to the Computer Security Problem. In: Abrams, Jajodia, and Podell, ed. *Information Security: An Integrated Collection of Essays*, Los Alamitos, CA: IEEE Computer Society Press, pp. 11--39.

10. Brinkley, D.L, Schell, R. R. (1995b) Concepts and Terminology for Computer Security. In: Abrams, Jajodia, and Podell, ed. *Information Security: An Integrated Collection of Essays*, Los Alamitos, CA: IEEE Computer Society Press, pp. 40--97.

11. Denning, D. E. (1982). *Cryptography and Data Security*. Reading, MA: Addison-Wesley Publishing, 1982.

12. Denning D. E. (1999). *Information Warfare and Security*. Reading, MA: Addison-Wesley Longman, 1999, pp 131--161.

13. Denning, P. J. (1990). ed. *Computers Under Attack: Intruders, Works and Viruses*. New York: ACM Press.

14. Gasser, M. (1988). *Building a Secure Computer System*. New York: Van Nostrand Reinhold, 1988.

15. Gligor, V. (1983). A Note on the Denial of Service Problem. Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, May, pp 139--149.

16. Hoffman, L. (1990). ed. *Rogue Programs: Viruses, Worms and Trojan Horses*. New York: Van Nostrand Reinhold.

17. Jelen, G. F. Williams, J. R. (1998). A Practical Approach to Measuring Assurance. in Proceedings of Fourteenth Computer Security Applications Conference, Phoenix, AZ, December, pp 333--343.

18. Kahn, (1996). *The Codebreakers: The Comprehensive History of Secret Communications from Ancient Times to the Internet*. 2nd ed., New York, NY: Scribner.

19. Karger, P. A., Schell, R.,R. (1974). Multics Security Evaluation: Vulnerability Analysis. ESD-TR-74-193. Vol II. Information Systems Technology Application Office Deputy for Command and Management Systems Electronic Systems Division (AFSC), Hanscom AFB, MA, 01730, 1974.

20. Karger, P. A., Schell, R. R., (2002). Thirty Years Later: The Lessons from the Multics Security Evaluation, Proceedings of the Annual Computer Security Applications Conference, Las Vegas, NV, December 2002, pp. 119-126.

21. Kaufman, C., Perlman, R., Speciner, M. (1995). *Network Security, Private Communication in a Public World*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1995.

22. Lack L. (2003). Using the Bootstrap Concept to Build an Adaptable and Compact Subversion Artifice, Masters Thesis, Naval Postgraduate School, Monterey, California, June 2003.

23. Lampson B. A (1973). Note on the Confinement Problem. *Comm. A. C. M.*, 16(10): 613--615.

24. Landwehr, C. E., Bull, A. R., McDermott, J. P., Choi, W. S. (1994). A Taxonomy of Computer Program Security Flaws. *ACM Computing Surveys*, 26: 211--254.

25. Levenson, N. (1995). Safeware. Reading, MA: Addison Wesley.

26. Levin, T., Padilla, S., Irvine, C. (1989). A Formal Model for UNIX SETUID. Proceedings of the 1989 IEEE Symposium on Security and Privacy, Oakland, CA, May, pp 73--83.

27. Lipner, S. (1982). Non-Discretionary Controls for Commercial Applications. Proceedings of the 1982 IEEE Symposium on Security and Privacy, Oakland, CA, April, pp 2--10.

28. Lunt, T. F. (1989). Access Control Policies: Some Unanswered Questions. *Computers and Security*, 8: 43--54.

29. Lunt, T. F. (1993). A Survey of Intrusion Detection Techniques. *Computers and Security*, 12: 405--418.

30. Murray, J. (2003). An Exfiltration Subversion Demonstration, Masters Thesis, Naval Postgraduate School, Monterey, California, June.

31. Myers, P. (1980). Subversion: The Neglected Aspect of Computer Security. Masters Thesis, U.S. Naval Postgraduate School, Monterey, California.

32. National Computer Security Center (1994). Introduction to Certification and Accreditation. NCSC-TG-029, National Computer Security Center, 9800 Savage Road, Fort George G. Meade, MD 20755-6000, January.

33. National Institute of Standards and Technology (1999). Common Criteria for Information Technology Security Evaluation, Version 2.1, CCIMB-99-031, http://csrc.nist.gov/cc/, August. (accessed December 2003)

34. Neumann, P. G. (1973). *Computer Related Risks*. New York: ACM Press.

35. Pfleeger, C. P. (1996). *Security in Computing*. 2nd ed. Englewood Cliffs, NJ: Prentice Hall, Inc.

36. Porras, P. A. (1992). STAT A State Transition Analysis Tool for Intrusion Detection, Masters thesis, University of California, Santa Barbara, CA.

37. Rogers, D. A. (2003). Framework for Dynamic Subversion, Masters Thesis, Naval Postgraduate School, Monterey, California, June.

38. Ritchie, D. M., Thompson, K. (1974). The Unix Time-Sharing System. *Comm. A. C. M.*, 17(7): 365--376.

39. Saltzer, J. H., Schroeder, M. D. (1975). The Protection of Information in Computer Systems. *Proceedings of the IEEE*, 63(9): 1278--1308.

40. Schneier, B. (1996). *Applied Cryptography*. 2nd ed. New York, NY: John Wiley.

41. Shannon, C. E. (1948). Mathematical Theory of Communication, *Bell System Tech. J.*, 27(4): 379--423, 623--656, 1948.

42. Shannon, C. E. (1949). Communication Theory of Secrecy Systems, *Bell System Tech. J.*, 28(4): 656-715.

43. Smith, R.E. (1997). Internet Cryptography. Reading, MA: Addison Wesley, 1997.

44. Stallings, W. (1998). Cryptography and Network Security Principals and Practice. 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1998.

45. Sterne, D. (1991). On the Buzzword "Security Policy". Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, CA, May, pp 219--230.

46. Stoll, C. (1989). *The Cuckoo's Egg*. New York: Doubleday.

47. Summers, R. (1997). *Secure Computing*. New York: McGraw Hill.

48. Thompson, K. (1984). Reflections on Trusting Trust. *Comm. A. C. M.* 27: 761--763.

49. Winkler, I. (1997). *Corporate Espionage*. Rocklin , CA: Prima Publishing.

50. Wolf, D. and Wolf, A. (2003). *The Easter Egg Archive*. http://www.eeggs.com/ (accessed December 2003)