Computer Security

Richard A. Kemmerer

INTRODUCTION

Computer security is an area that is growing in importance as more business applications are being automated and more vital and sensitive information is being stored in computers. Almost daily one can read newspaper accounts of computer abuse. Stealing information from a computer (e.g., a competitor's bid on a lucrative contract) is just like stealing goods from a warehouse, for information is a commodity. The primary difference is that the loss of information often goes unnoticed.

The term *computer security* means the protection of resources (including data and programs) from unauthorized disclosure, modification or destruction. In addition, the system resources must also be protected (i.e., access to system services should not be denied). These computer security properties are usually referred to as confidentiality, integrity, and availability. More precisely:

Confidentiality ensures that sensitive information is not disclosed to unauthorized recipients.

Integrity ensures that data and programs are modified or destroyed only in a specified and authorized manner.

Availability ensures that the resources of the system will be usable whenever they are needed by an authorized user.

The degree to which each of these three properties is needed varies from application to application. For instance, the defense industry is primarily interested in confidentiality. In contrast, the banking industry is primarily interested in integrity, and the telephone industry may value availability most. The exact requirements that are needed for a particular system or application are expressed in the security policy for that system or application.

BACKGROUND

In the early days of computing when standalone systems were used by one user at a time, computer security consisted primarily of physical security. That is, the computer and its peripherals were locked in a secure area with a guard at the door that checked each user's identification before allowing them to enter the room.

As time sharing systems emerged in the mid to late 1960s and multiple jobs and users were able to run at the same time, controlling the access to the data on the system became a major point of concern. One solution that was used was to process classified data one level at a time and "sanitizing" the system after the jobs from one level were run and before the jobs for the next level were run. This procedural approach to computer security was known as *periods processing* because the jobs for each level were all run in their particular period of the day. This was an inefficient way to use the system, and an effort was made to find more efficient software solutions to the multilevel security problem.

The initial interest in computer security was spearheaded by the U.S. Department of Defense (DoD). The "Ware Report," which was published in 1970 [War 70], pointed out the need for computer security and highlighted the difficulties in evaluating a system to determine if it provided the necessary security for particular applications. This report was the final report of the Defense Science Board Study, and was named after Willis Ware of the RAND Corporation who chaired the advisory group.

In 1972 the Air Force Electronics System Division sponsored the Computer Technology Planning Study. The intent of this study was to define the Research and Development paths required to make secure computers a reality in the USAF. The final report from this study was called the "Anderson Report" [And 72]. The major contribution from this study was the *reference monitor* concept, which led to security kernel architectures.

In the mid to late 1970s a number of systems were designed and implemented using a security kernel architecture. These were built to run on PDP-11s [Mil 76, MD 79, WKP 80], MULTICS [SCS 77], Honeywell Level 6 [Fra 83], and IBM 370s [GLP 79]. The systems were all sponsored by the defense establishment, and as a result, they were concerned primarily with confidentiality.

Another effort that occurred in the mid to late 1970s was the use of "tiger teams" to test the security of a system. These teams attempted to obtain unauthorized access to the system by exploiting design and implementation errors [Lin 75, HGB 80]. The tiger team studies further showed the difficulty of providing secure software; virtually every system that was attacked by a tiger team was penetrated.

In 1977 the DoD Computer Security Initiative began. This was an effort to consolidate the results of the work of the last decade and to focus on the computer security issues that had been raised. One of the results of the initiative was a number of workshops and conferences on computer security topics. An effort was also made to interest the vendor community in building secure systems.

In 1981 the DoD Computer Security Center (CSC) was established. Its charter was to evaluate commercially developed computer systems with regard to computer security, to set standards for the evaluations, and to conduct computer security research and development. The center was established at the National Security Agency (NSA) and it dealt primarily with the security problems of the defense community.

In 1983 the DoD Computer Security Center released its first official evaluation criteria, titled the "Department of Defense Trusted Computer System Evaluation Criteria" [DoD 83]. This book, which has an orange cover, is usually called the "orange book" or the "TCSEC." This effort was the first attempt to provide a set of criteria for evaluating the effectiveness of the security controls of a computer system. In 1985, after some minor revisions, the orange book was made a DoD standard. In 1985 the CSC was renamed the National Computer Security Center (NCSC), although it still dealt primarily with defense problems.

In 1987 the Computer Security Act partitioned computer security in the U.S. Federal government into two distinct branches. The National Computer Security Center at NSA was to deal with classified information, and the National Computer Systems

Laboratory (NCSL) at the National Institute of Standards and Technology (NIST) was to deal with unclassified information. In addition, NIST was tasked to support the civil government and unclassified defense communities. This act of Congress clearly pointed out the need for computer security in the non-defense communities.

The Canadian government also recognized the need for an evaluation capability and formed the Canadian System Security Centre (CSSC) in 1988. In May 1989 the first version of the Canadian Trusted Computer Product Evaluation Criteria (CTCPEC or Canadian Criteria) was released. The most recent version of the Canadian Criteria (version 3.0e) was released in January 1993 [CSSC 93].

The need for computer security in the non-defense community was also realized by European nations. In 1989 the West German Information Security Agency published a criteria for the evaluation of the trustworthiness of information technology systems [ZSI 89]. Unlike the TCSEC, which dealt primarily with confidentiality, the German criteria also addressed the issues of integrity and availability and of network security. At the same time the British Commercial Computer Security Center (CCSC), which was established by the Department of Trade and Industry and the Ministry of Defense, also released a computer security criteria [DTI 89]. A notable feature of the British criteria is that it provided a "claims language" that allowed vendors to construct security-worthiness statements about their products.

In 1990 the countries of France, Germany, the Netherlands, and the United Kingdom made a joint effort to produce an international set of computer security evaluation criteria. The document was called the "Information Technology Security Evaluation Criteria" (ITSEC) [ITS 90]. This "harmonized" effort was primarily a combination of the evaluation classes contained in the German criteria and the claims language from the British criteria.

In 1990 a National Research Council (NRC) study group, the System Security Study Committee, pointed out the need for computer security in the commercial sector. The final report from this committee - "Computers at Risk: Safe Computing in the Information Age" - alerted the public to the risks posed by inadequate security and proposed actions to be taken to solve the problem [NRC 91]. Most noteworthy of their proposals was the establishment of an Information Security Foundation (ISF), which would be a private not-for-profit organization. One of the tasks proposed for the ISF was to produce a comprehensive set of Generally Accepted System Security Principles (GSSP) that would be a basic set of principles for designing, using and managing secure systems. The ISF would also be responsible for certifying commercial security systems, much like the Underwriters Laboratory certifies electrical systems. The NRC committee also recommended that attempts be made to harmonize with the international criteria efforts.

In an attempt to broaden the scope of the TCSEC the U.S. Federal government started a joint program between NIST and NSA called the Federal Criteria (FC) project. In December of 1992 the first draft of the FC [FC 92] was circulated for review. The final FC document was originally intended to be a Federal Information Processing Standard (FIPS) for use by the U.S. Federal government, which would replace the TCSEC.

In June 1993, at the start of a joint NIST/NSA workshop that was held to discuss the draft FC, the participants were informed that NIST and NSA had decided to work with the Canadians and the Europeans to develop a "Common Information Technology Security Criteria" (CC). A Common Criteria Editorial Board (CCEB) was also created with members from Canada, France, Germany, the United Kingdom and the United States. The CCEB was tasked with aligning the TCSEC, ITSEC and the CTCPEC. Later a Common Criteria Implementation Board (CCIB)was created to develop the next generation of Information Technology security evaluation criteria.

Version 1.0 of the Common Criteria was produced in January 1996. This version underwent an extensive trial evaluation, was put out for comments, and after an extensive revision process, resulted in the release of version 2.0 in May of 1998 [CC 98]. After this version was released, the International Organization for Standardization (ISO) began working with the CCIB to make the CC an international standard. On December 1, 1999 the ISO approved and published what was eventually the CC text as a new International Standard, ISO 15408. The standard had some minor changes from CC version 2.0, and these changes were incorporated into the CC, resulting in CC version 2.1 [CC 99]. During this time another important event that occurred was that in October 1998 Canada, France, Germany, the United Kingdom, and the United States signed a Mutual Recognition Arrangement (MRA) for Common Criteria based evaluations, and in May 2000 a second MRA, which included the additional countries of Australia, New Zealand, Finland, Greece, Italy, the Netherlands, Norway, and Spain, was signed. The mutual recognition provided by this agreement means that a Common Criteria certificate awarded in one country is recognized in all member countries. This was a significant step forward.

COMPUTER SECURITY TODAY

Computer security consists largely of defensive methods used to detect and thwart would-be intruders. The principles of computer security thus arise from the kinds of threats intruders can impose. For instance, one general security stance is that "everything that is not permitted is prohibited." If this principle is enforced, then an intruder can not get access to some object just because the security administrator did not consider whether it should be restricted or not. Many members of the security community believe that if software were designed with more of an emphasis on security and if systems were configured properly, then there would be fewer security problems.

This section first presents four approaches to achieving secure computing, which may be used used to implement an organization’s security policy. The last section then compares and contrasts today’s approaches to developing secure systems with standard software development techniques.

Approaches to Secure Computing

There are four general approaches to achieving a secure computing environment. They are the use of special procedures for working with the system, the inclusion of additional functions or mechanisms in the system, the use of assurance techniques to increase one’s confidence in the security of the system, and the use of intrusion detection systems. Each of these is discussed in the following subsections.

Some security requirements can either be achieved by requiring procedures to be followed or by using system mechanisms or functions to enforce the requirement. Also, in some cases system users need to follow specific procedures in order to make security mechanisms effective. It is also possible to trade off assurance techniques for less mechanism. For instance, one can use assurance techniques, like formal proofs, to assure that the system can not get into a particular state; thus, alleviating the need for the software mechanism that would deal with that state.

Procedural Approaches

Procedural approaches prescribe the appropriate behavior for the user to follow when using the system. The periods processing approach for processing jobs at different security levels is an example of a procedural solution to satisfy a security requirement.

Many successful penetrations are initiated by an intruder guessing a user's password. With the advent of personal computers and dial-up modems this has become much more of a problem. In addition, the availability of online dictionaries also makes the process of guessing easier. Would-be penetrators also have lists of commonly used passwords that they can try automatically with the aid of their personal computer, and if passwords are short they are easily found by an exhaustive search. In a study carried out at Bell Labs in 1979 [MT 79] it was shown that by using the available computer power of the day an exhaustive search could be used to try all four letter lower case passwords in 7 minutes. If the characters were mixed case and numeric characters the search took 5 hours, and if the characters of the password were chosen from all printable characters the search took 28 hours. For six character passwords the respective search times were 107 hours, 2.2 years and 29 years. Using the desktop workstations that are readily available today, these times can be reduced by a factor of 100. Thus, a four letter lower case only password could be exhaustively searched in less than a minute and a six letter lower case only password in less than an hour. Another vulnerability with passwords is that computer vendors deliver systems with standard accounts that have default passwords, and these passwords often are not changed by the system administrators.

User guidelines for the appropriate choice of a password is the most prevalent example of using procedural approaches to achieve a security requirement. For example, to deter password guessing by a potential intruder one should choose a long password (at least eight characters) that is not obvious and not easily guessable (e.g., not a spouse's first name, a middle name, a login name, or any of these spelled backwards). Passwords should also use both upper and lower case letters, numerics and possibly special symbols. In addition, a password should not be written down, or if it is it should not be written in an obvious place. Furthermore, users should be trained to change their passwords at appropriate intervals. Many of these restrictions can be enforced by the system when a user chooses a new password (See the Authentication Mechanisms section below.).

Another example of a procedural approach is a set of rules for the appropriate handling of removable storage devices. Oftentimes data that is perfectly safe while in a protected system is compromised by a penetrator getting access to removable storage, such as a dump tape, and analyzing it on an unprotected system. For this reason most security conscious organizations have explicit rules for handling removable media, such as requiring them to be stored in an approved vault.

Functions and Mechanism

Including additional functions or mechanisms in a computer system is another way of enhancing computer security. The mechanisms presented in this section are grouped into authentication mechanisms, access control, and inference control.

Authentication Mechanisms -- Authentication mechanisms are used to assure that a particular user is who he/she claims to be. The first mechanism discussed is the *secure attention key*. This key, when hit by a user at a terminal, kills any process running at the terminal except the true system listener and thus guarantees a trusted path to the system. This will foil attempts at "spoofing," which is the process of fooling a user into believing that he/she is talking to the system, resulting in information being revealed. For instance, the spoofer may display what looks like the system login prompt on a terminal to make the terminal appear to be idle. Then when an unsuspecting user begins to use the terminal, the spoofer retrieves the login name and asks for the user's password. After obtaining this information, the spoofer displays a message to try again and returns ownership of the terminal to the system. If a secure attention key is used, it is important that users make a habit of always hitting the key to begin a dialogue with the system. One way of ensuring this is for the system to only display the login prompt after the key is depressed. This is an example of procedures and mechanism working together to achieve a security property.

Most of the password guidelines that were discussed above as a procedural approach can be enforced by the system. For instance, the password program can require long passwords and it can check the password chosen against an online dictionary or against a list of obvious passwords. The login program can also inform the user that it is time to change passwords and not allow further logins if the password is not changed in time. Finally, the system can generate secure passwords for the users using a secure password generator.

Password files stored in the system may be compromised like any other file. Therefore, it is not good practice to store passwords in the clear. Instead, a *one-way function* (i.e., a function whose inverse is computationally infeasible to determine) is used to encipher passwords and the result is stored in the password file. When a user's password is presented at login time it is enciphered and compared to the stored value. By using one-way functions to encipher passwords the login file can be made public.

One of the problems that has occurred when using passwords for remote access is that an attacker can obtain a user's password by "sniffing" the network traffic. To thwart this attack many systems are now using one-time passwords. This is based on the code book approach used by spies during world war II. The password code book contains a list of passwords that are used one time and then are never used again. Each time that a user wants to be authenicated he/she looks up the next password in the code book.

By using a mathematical algorithm to generate the list, the list can be virtual and stored as a program in a credit card type of device. These devices are usually referred to as token cards. Other token cards are time sensitive. That is, they generate the password by using a calculation that is based on the current time. Still other token cards use a challenge response-approach. With this approach when a user contacts a remote machine to be authenticated the machine returns a unique number as a challenge. The user then enters his/her pin into the token card along with the challenge number, and the card returns a response, which the user then sends to the remote machine.

The most sophisticated authentication devices are those that depend on a unique physiological or behavioral characteristic that can be examined and quantified for each user. When a user wants to be authenticated his/her characteristics are sampled, converted into a digital form, and compared to the previously stored sample characteristics for that user. The most common biometric devices are based on fingerprints, handprints, or retina patterns. The most common behavioral devices use voice, signature, or keystroke characteristics. Most of these devices are used in a two-factor authentication system, such as with passwords.

Access Control -- Assuming that by using authentication mechanisms and good password practice the system can guarantee that users are who they claim to be, the next step is to provide a means of limiting a user's access to only those files that policy determines should be accessed. These controls are referred to as *access control*. Different applications and systems have different security requirements, and these requirements are expressed in the *access control policy* for the application or system. Access control policies are enforced by the access control mechanisms.

When describing access control policies and mechanisms it is necessary to consider the *subjects* and *objects* of the system. Subjects are the users of the system along with any active entities that act on behalf of the user or the system (e.g., user processes). Objects are the resources or passive entities of the system (e.g., files, programs, memory, devices). Subjects may also be objects (e.g., procedures). The *access control mechanism* determines for each subject what *access modes* (sometimes called access rights), such as read, write, or execute, it has for each object.

There are two types of access control: *discretionary access control* (DAC) and *mandatory access control* (MAC). More precisely:

Discretionary access control - the owner of an object specifies what type of access the other users can have to the object. Thus, access is at the discretion of the owner.

Mandatory access control - the system determines whether a user can access a resource based on the security attributes (e.g., labels or markings) of both the user and the object.

Mandatory access control is often called non-discretionary access control.

One of the earliest forms of discretionary access control, which is still being used on some mainframe and PC systems, is the use of passwords for file access. That is, the owner selects a password for each file and distributes the password to those users to whom the owner wants to give access to the file. A major problem with this approach is that one user may pass a password on to another user without the consent of the owner of the file. A second major problem is that the owner cannot revoke access from one user without revoking access from all users and then selecting a new password for the file. Another problem with this approach is that the passwords tend to get embedded in other files and are vulnerable to browsing.

Another form of DAC, which is used on UNIX systems, is the *owner/group/other* approach. With this approach the owner of a file assigns the types of access that are allowed for the owner, for all users in the group associated with the file, and for all users on the system. The problem with this approach is that if an owner wants to give access to a single user, he either has to setup a group with only that user in it or else give access to everyone on the system. That is, because a process running on behalf of a user is allowed to be a member of only a finite number of groups (usually sixteen), there is often no group that contains only the specified user(s) to which access should be given. As a result the owner needs to give access to more individuals than is desired.

A convenient way of describing access rights is with an *access matrix*. In the access matrix rows correspond to subjects and columns correspond to objects. Each entry in the matrix is a set of access rights that indicate the access that the subject associated with the row has for the object associated with the column. The following is an example access matrix. From this matrix one can determine that subject S3 has read and write access to object O2 and execute access to object O3.

	OBJECTS				
SUBJECTS	O1	O2	O3	O4	O5
S1	R		W	RW	W
S2		E		R	
S3		RW	E		
S4	RE		RW		RE

Example Access Matrix

Although an access matrix is a convenient way of describing the allowable accesses, it is not an efficient way of representing these accesses in a computer, because the matrix is usually sparse. There are two commonly used and more efficient ways of representing an access matrix in a computer system: *access control lists* (sometimes called authorization lists) and *capability lists* (often called c-lists). With the access control list approach each object has an access list associated with it. This list contains the name of each subject that has access to the object along with the modes of access allowed. In contrast the capability list approach associates a list with each subject. The elements of the list are *capabilities*, which can be thought of as tickets that contain an object name and the modes of access allowed to the object. A subject's capability list defines the environment or domain that the subject may directly access. The reader should note that an access list corresponds to a column in the access matrix and a capability list corresponds to a row.

An important aspect of either approach is that both the capabilities and the elements of access lists must be unforgeable or else the entire protection mechanism breaks down. One way of guaranteeing the unforgeability of these elements is by restricting access to them through an intermediary trusted piece of code. The reference monitor introduced in the Anderson report [And 72] is one such mechanism.

Access control policies often incorporate *access hierarchies*. That is, subjects may have different ranks ranging from the most to the least privileged, where the more privileged user automatically gets the rights of the less privileged user. For instance, in a UNIX system a subject with "superuser" privilege can access any object in the system. Multics systems provide eight hierarchical rings that separate the operating system from system utilities and users, and different level users from each other.

As an example of an access control policy that incorporates access hierarchies, consider the following mandatory control policy. In this model every subject and every object has an access class made up of a level (e.g., unclassified, confidential, and secret) and a (possibly empty) set of categories (e.g., crypto, nuclear, and intelligence). Levels are ordered and categories are not. When comparing access classes the result can be equal, less than, greater than, or not comparable. For instance, the access class with level secret and category set containing only crypto is greater than the access class with level unclassified and an empty category set. Furthermore, secret/{crypto} is less than secret/{crypto,nuclear}, and secret/{crypto} is not comparable to confidential/{nuclear}. The access rules for this policy are as follows. A subject may obtain read permission to an object if its access class is greater than or equal to the access class of the object. This is known as the *simple security property*. In addition, a subject may write an object if the subject's access class is less than or equal to the access class of the object. This is a simplified version of the **−property* (pronounced star property).

To assure that all access control policies are enforced a means of mediating each access of an object by a subject is needed. The *reference monitor* [And 72] provides this mediation. A reference monitor is defined by three basic properties.

1. It must be tamperproof. That is, it should be isolated from modification by system entities.
2. It must always be invoked. That is, it must mediate every access.
3. It must be small enough to be subjected to analysis and tests, the completeness of which can be assured.

A *security kernel* is defined as the hardware and software that realize the reference monitor. The idea is to keep the security kernel small and to have it contain the security relevant parts of the system.

Inference Controls -- The last class of security mechanisms discussed in this section is inference controls. These controls attempt to restrict database access to sensitive information about individuals while providing access to statistics about groups of individuals. The ideal is to have a statistical database that discloses no sensitive data.

As an example of the type of threat that is addressed by inference control mechanisms consider a database that contains enrollment and grade statistics for a university class. If Morgan is the only Economics major in a particular class one could deduce Morgan's grade by retrieving the average grade for the course, the average grade of all non-economics majors in the class, and the number of students in the class.

Two approaches to solving the inference problem are to restrict queries that reveal certain types of statistics and to add "noise" to the results returned. To foil small and large query set attacks, such as the Morgan example above, a technique called *query-set-size control* is introduced. This forbids the release of any statistics pertaining to a group less than some predetermined size n or greater than N−n, where N is the total number of records in the database. Other techniques, restrict queries with more than some predetermined number of records in common or with too many attributes specified.

Among the techniques that add noise to the statistical results returned are *systematic rounding, random rounding,* and *controlled rounding*. The third alternative requires the sum of rounded statistics to equal their rounded sum. The idea is that it is all right if the user posing the queries knows the exact answer about a large sample, but nothing should be released about a small sample. *Random sample query control* is another promising approach to solving the inference problem. With this approach each statistic is computed using 80-90 per cent of the total number of records and a different sample is used to compute each statistic. For an excellent presentation of these techniques see [Den 82]. For other information on database security see the section in this book.

Assurance Techniques

The third approach to enhancing the security of a system is to subject the system to rigorous assurance techniques that will raise one's confidence that the system will perform as desired. Among these techniques are penetration analysis, formal specification and verification, and covert channel analysis. None of these methods guarantee a secure system. They only increase one's confidence in the security of the system.

Penetration Analysis -- One approach to locating security flaws in computer systems is *penetration analysis*, This approach uses a collection of known flaws, generalizes these flaws, and tries to apply them to the system being analyzed. Usually a team of penetrators, called a tiger team, is given the task of trying to enter the system. Flaws in many major systems have been located by using this approach [HGB 80, Lin 75].

The problem with the tiger team approach is that like testing, "penetration teams prove the presence, not absence of protection failures" [Pop 74]. This observation has led to the use of *formal specification* and *verification techniques* to increase ones confidence in the reliability and security of a computer system.

Formal Verification -- Formal verification demonstrates that an implementation is consistent with its requirements. This task is approached by decomposing it into a number of easier problems. The critical requirements, which are usually a natural language statement of what is desired, are first stated in precise mathematical terms. This is known as the *formal model* or *critical criteria* for the system. For example, the formal model for a secure system could be that information at one security level does not flow to another security level. Next, a high level formal specification of the system is stated. This specification gives a precise mathematical description of the behavior of the system omitting all implementation details, such as resource limitations. This is followed by a series of less abstract specifications each of which implements the next higher level specification, but with more detail. Finally, the system is coded in a high order language. This high order language implementation must be shown to be consistent with the original requirements.

It should be emphasized that demonstrating that high order language code is consistent with security requirements is a difficult process. The process is made tractable by verifying the design at every step. The first step of the verification process is to informally verify that the formal model properly reflects the security requirements. This is the only informal step in the process. Since the formal model is at a high level of abstraction and should contain no unnecessary details, it is usually a simple task to review the formal model with the customer who generated the requirements and determine whether the model properly reflects the critical requirements. Next, it is necessary to prove that the highest level specifications are consistent with the formal model. Both a state machine approach [Hoa 72] and an algebraic approach [GHM 78] are possible.

After the highest level formal specification has been shown to be consistent with the formal model it is necessary to show that the next lower level specification, if one exists, is consistent with the level above it. This process continues from level to level until the lowest level specification is shown to be consistent with the level above it. Finally, it is necessary to show that the high order language implementation is consistent with the lowest level specification. By transitivity, the implementation is thus shown to be consistent with the formal model. For a detailed description of the formal specification and verification process for a secure system see [Kem 90].

The advent of the security kernel as a means of encapsulating all security relevant aspects of the system makes formal verification feasible. That is, by developing kernel architectures that minimize the amount and complexity of software involved in security decisions and enforcement, the chances of successfully verifying that the system meets its security requirements are greatly increased. Because the remainder of the system is written using the facilities provided by the kernel, only the kernel code must be verified. Examples of work in this area are [MD 79, WKP 80, Sil 83, Bev 89]

Covert Channel Analysis -- Secure computer systems use both mandatory and discretionary access controls to restrict the flow of information through legitimate communication channels such as files, shared memory and process signals. Unfortunately, in practice one finds that computer systems are built such that users are not limited to communicating only through the intended communication channels. As a result, a well-founded concern of security-conscious system designers is the potential exploitation of system storage locations and timing facilities to provide unforeseen communication channels to users. These illegitimate channels are known as covert storage and timing channels [Lam 73, Lip 75, Mil 76, Kem 83].

Covert channels signal information through system facilities not intended for data transfer, and they support this communication using methods not detected or regulated by the access control mechanisms. *Storage channels* transfer information using the manipulation of storage locations for their coding scheme. That is, the sending process alters some system attribute (e.g., a file lock or a device busy flag) and the receiving process monitors the alteration. For example, if two processes have only write access to a file, then the sending process could lock the file, and the receiving process could detect this change by attempting to write to the file. In this way, the receiver could interpret the file being locked as a one and it not being locked as a zero. *Timing channels* transfer information using the passing of time for their coding scheme; the sending process modulates the receiver's response time to detect a change in some shared entity.

Although there is concern that a user at a high security level may use a covert channel to signal information to a user at a lower level, the major threat from a covert channel is its potential to be employed by a Trojan horse. A *Trojan horse* is a program that gives the appearance of providing normal functionality, but whose execution results in undesired side effects.

The severity of a covert channel threat has been traditionally measured in terms of the channel's bandwidth, i.e. the number of bits signaled per second. The higher the bandwidth, the greater the potential for serious compromise. An alarming possibility regarding covert channels is that as operating systems are ported to faster hardware architectures, the bandwidths of their covert channels may increase significantly. In fact, timing channels with estimated bandwidths in the megabits per second range have been demonstrated on symmetric multi-processing architectures [COV 90].

In addressing the threat of covert channels two major challenges have been identified. The first challenge is in developing techniques to identify covert channels in a comprehensive, systematic manner. Several covert channel analysis techniques have been proposed and utilized. Usually these techniques base their analysis either on code inspection or on inspection of the high level specification. Among these techniques are the Non-Interference approach [GM 82], the Shared Resource Matrix (SRM) methodology [Kem 83], and Information Flow analysis [Den 76]. The second, and more difficult challenge, is in removing the channels, or at least lowering their bandwidths, without rendering the system unacceptably slow or restrictive. References [Hu 91, KW 91] provide excellent examples of how this second covert channel challenge is being met.

Prior to the 1980s the covert channel analysis that took place was mostly *ad hoc*. The SRM approach, which was introduced in the early 1980s did not aid in the analysis of covert channels, but rather gave a form for describing what attributes might be used for signaling information. More importantly, it identified those attributes that could not be used. This gave the analyst more time to concentrate on those that could be used.

In the mid 1980s some researchers began applying the non-interference approach to systems. With this approach the failed proofs of the unwinding theorems should lead the analyst to the flows to consider, but like the SRM it too did not aid the analyst in the actual analysis. With both the SRM and the non-interference approaches the analyst had to come up with the signaling sequences and determine whether they could be used as covert channels [HKM 87].

A more recent approach called the Covert Flow Tree (CFT) approach [KP 91] uses tree data structures to model the flow of information from one shared attribute to another. The CFTs are then used to perform systematic searches for operation sequences that allow information to be relayed through attributes and eventually detected by a listening process. When traversed, the paths of a CFT yield a comprehensive list of operation sequences that support communication via a particular resource attribute. These operation sequences are then analyzed and either discharged as benign or determined to be covert communication channels. That is, the analyst with his/her experience is still the one that makes the determination.

Intrusion Detection and Prevention

Over the past decade, significant progress has been made toward the improvement of computer system security. Unfortunately, the undeniable reality remains that all computers are vulnerable to compromise. Even the most secure systems built today are vulnerable to authorized users who abuse their privileges. Given this reality, the need for user accountability is very important. Accountability is key, both as a deterrent and for terminating abusive computer usage once it is discovered. In addition, the need to maintain a record of the transactions that have occurred on a system is crucial for performing damage assessment. In recognition of these needs and in recognition that security violations are a fact of life for computers, many systems implement a form of transaction record keeping called *audit collection*. The data collected is called an *audit log*. Additional systems and/or software are often required to generate the necessary audit data. For example, in order to produce audit records for Sun Microsystem's Solaris one needs to use the Basic Security Module (BSM) [Sun 91].

Although the audit logs normally contain enough information to determine that a computer abuse has occurred, the amount of data collected is so voluminous that manual audit data analysis is impractical. By introducing automated audit data analysis tools, referred to as *intrusion detection systems* (IDSs), security violations that might have once gone unnoticed can be identified. In addition, when intrusion detection is performed in real-time the audit data can be used to track a user's behavior and to determine if the user's current actions represent a threat to security. If they do, the system can halt the user's actions.

Within the past ten years, there has been a steadily growing interest in the research and development of intrusion detection systems. Historically detection has been achieved following two different approaches: anomaly detection and misuse detection. Anomaly detection relies on models of the ‘‘normal’’ behavior of a computer system, while misuse detection takes a complementary approach. Misuse detection tools are equipped with a number of attack descriptions. These descriptions (or "signatures") are matched against the stream of audit data looking for evidence that a modeled attack is occurring. Misuse and anomaly detection both have advantages and disadvantages. Misuse detection systems can perform focused analysis of the audit data and usually they produce only a few false positives. At the same time, misuse detection systems can detect only those attacks that have been modeled. Anomaly detection systems have the advantage of being able to detect previously unknown attacks. This advantage is paid for in terms of the large number of false positives and the difficulty of training a system with respect to a very dynamic environment. An introduction and overview of different intrusion detection approaches can be found in [Amo 99, Bac 00, BM 01, NNM 01]. Surveys of implemented intrusion detection systems, many of which are in operation today, can be found in [Lun 88, MSW 90, Neu 90, NSS 00]. This section briefly reviews the current approaches to intrusion detection.

Statistical Anomaly Detection -- One of the most prevalent approaches used in the development of intrusion detection systems involves the use of statistical analyses to measure variations in the volume and type of audit data produced on a system. The statistical analysis of audit data can be applied to individual user audit trails or applied to all of the target system's audit records as a whole. There are two techniques for intrusion detection using statistical anomaly detection: *threshold detection* and *profile-based anomaly detection*.

The idea of threshold detection is to record each occurrence of a specific event and, as the name implies, detect when the number of occurrences of that event surpass a reasonable amount that one might expect to occur during normal system operation. The event is such that an unnaturally high number of occurrences within a specified period of time may indicate the presence of an intruder.

Profile-based anomaly detection uses statistical measures to identify intrusions by monitoring a system's audit logs for usage that deviates from established patterns of usage. These focus on the users, on the applications, or on the network. The main advantage of anomaly detection is that it provides a means of detecting intrusions, without *a priori* knowledge of the security flaws in the target system. The IDES system uses a profile-based anomaly detection component and a rule-based component for identifying known penetrations [Lun 90]. A later version of the IDES system, called NIDES, also uses profile-based anomaly detection [JV 94].

Rule-Based Anomaly Detection -- Rule-based anomaly detection is like statistical anomaly detection except that rather than using statistical formulas to identify normal usage patterns in audit data, rule-based anomaly detectors use sets of rules to represent and store normal usage patterns. These rules are then used to pattern match sequences of audit records to the expected audit trails to see if they represent normal or anomalous behavior. An example intrusion detection implementation that employs rule-based anomaly detection is Wisdom and Sense (W&S) [VL 89]. Specification-based and immunology-based detection systems are similar to rule-based anomaly detection systems. Information on these systems can be found in [FHSL 96, KRL 97, SU 99].

A simple but quite effective anomaly based tool is tripwire [KS 93], which is a program that monitors particular attributes of files to track changes. A configuration file specifies the particular files or directories to be monitored and what attributes to monitor. Although tripwire started as a free UNIX tool, there are now commercial versions for Windows NT and most versions of UNIX and UNIX-like systems.

Rule-Based Misuse Detection -- A rule-based misuse detection system (sometimes called a rule-based penetration identification system) is an expert system whose rules fire when audit records are parsed that appear to indicate suspicious, if not illegal, user activity. The rules may recognize single audited events that represent significant danger to the system by themselves, or they may recognize sequences of events that represent an entire penetration scenario.

Rule-based penetration identifiers have become a common supplemental component of intrusion detection systems that employ other approaches. For example, intrusion detection systems that employ anomaly detectors often supplement their detection capabilities with expert rules for identifying known penetration. The expert rules offer the additional capability of identifying dubious behavior, even when the behavior appears to conform with established patterns of use. Example misuse detection systems are USTAT [IKP 95] and P-BEST [LP 99]. Examples of intrusion detection implementations that supplement their anomaly detection components with expert penetration rules include IDES and W&S. Unlike anomaly detection systems, the rule-base of a penetration identifier is very machine specific.

Static Audit Tools -- Efforts to develop practical tools for preventing intrusions are also available. One of the first prevention tools was the Computer Oracle Password and Security System (COPS), developed by Farmer and Spafford [FS 90]. COPS was designed to aid UNIX system administrators in testing their configurations for common weaknesses often exploited by intruders. COPS is a collection of shell scripts and programs that search out and identify a myriad of security holes (e.g., writable system files) commonly present on UNIX systems. More recent static audit tools include the ISS System Security Scanner and the Kane Security Analyst.

Essentially, no intrusion detection approach stands alone as a catch-all for computer penetrations. Instead, each approach is technically suited to identify a subset of the security violations to which a computer system is subject. Accordingly, intrusion detection system designers often implement multiple approaches within a single intrusion detection system.

Computer Security and Software Engineering

The key difference between secure software and other high quality software is that secure systems have to be able to withstand active attacks by potential penetrators. When developing any reliable software one must try to minimize the faults in the system and assure that accidental abnormal user actions or abnormal external events do not result in undesired events. When developing a secure system the developers must also assure that intentional abnormal actions cannot compromise the system. That is, secure systems must be able to avoid problems caused by malicious users with unlimited resources.

Because security is a system requirement just like performance, capability, and cost, it must be designed in from the beginning. It must not be added on after-the-fact. In addition, because security is only one of many goals for a system, it may be necessary to trade off certain security requirements to achieve other goals, such as user friendliness.

When designers first start thinking about developing secure systems they often think that it would be beneficial to keep the system design secret. However, the security community realized many years ago that the benefits of an open design far outweigh any advantages of keeping the design hidden from would be intruders. The *open design principle* [SS 75] states that "the mechanism should not depend on the ignorance of potential attackers..." By having peer reviews of the design of a secure system, security weaknesses in the system are often discovered during the design phase. It is always better to have a colleague find a weakness during the design phase rather than having it discovered through a compromise after the system has been fielded.

The main difference between secure systems and other high quality systems is that secure systems are subject to malicious attack; therefore, it should be no surprise that a primary difference in developing secure systems is the need for additional testing and for a somewhat different form of testing. Penetration analysis is a form of testing. However, unlike standard testing techniques where a tester's primary task is to demonstrate that the program gives the correct result when presented with varying inputs, with penetration analysis the system is given input that is not expected to give a good result or operations are executed in an unexpected sequence. Thus, instead of concentrating on showing that the system gives the expected result when used properly, the testing concentrates on demonstrating that the system gives an undesired result when used improperly. Thus, the test cases concentrate more on trying the unusual or the unexpected.

Finally, because the need for a high level of assurance was recognized by the security community many years ago, the use of formal methods is more prevalent in the design and analysis of secure systems than in most other software development areas. All of the national and international evaluation criteria specify the need for the use of formal methods to achieve added assurance.

FUTURE TRENDS

Advances in computer security in the next decade will be in the areas of computer networks, privacy issues, trusted systems, and education. The growth of the Internet and the World Wide Web (WWW) during the past few years has been phenomenal. The Internet is currently serving tens of millions of people connected through millions of computers. Most every business and government institution has a web page, and the web and web browsing are fast becoming the primary source of information for people of all ages. As computers are becoming more prevalent and are linked together on world-wide networks there is more concern about network security. In addition, as more personal data is being kept in databases there is a growing concern about the security of that data. Also, with the increased use of computers in life-critical systems there is the broadening of the concerns of security to include safety and dependability. Finally, there are not enough qualified computer security personnel to handle the technical problems that arise. Each of these is discussed in more detail in the following paragraphs.

Network Security

As more applications are distributed and the use of networking increases there is also an increased need for network security. Because networks can be viewed as systems, they have most of the same problems that have been discussed for computer security, such as authentication, access control, and availability. However, these problems become more complex as the applications become distributed. For instance, spoofing across the network is harder to foil, because it deals with "mutually suspicious subsystems." That is, the remote system or server needs to authenticate that the user that is logging on or requesting resources is who he/she claims to be. However, the user also needs to verify that he/she is connected to the correct system, even though that system may be thousands of miles away. With networked systems there is also an increase in the number of points of attack. In 1988 a National Research Council study raised concern about the present state of network security [NRC 89].

Currently most messages on networks, like the Internet, are unencrypted. However, in the future, as more sensitive data is transmitted over these networks, more encryption will be needed. When considering a secure network that uses encryption to achieve its security one must consider both encryption algorithms and encryption protocols. An *encryption algorithm*, such as DES or RSA, is used to convert clear text into cipher text or cipher text into clear text. That is, the unencrypted message (clear text) is enciphered using a particular encryption algorithm to produce the unreadable cipher text. Similarly, the same or a symmetric algorithm is used to recover the original clear text message from the encrypted cipher text. An *encryption protocol* is a set of rules or procedures for using the encryption algorithm to send and receive messages in a secure manner over a network. Recently there has been an interest in encryption protocols, and a number of protocols have been shown to be insecure. This interest is expected to increase in the next decade. The National Institute of Standards and Technology has recently announced the selection of a new encryption algorithm as the proposed standard for use by U.S. Government organizations to protect sensitive (unclassified) information. The Advanced Encryption Standard (AES) will be a new Federal Information Processing Standard (FIPS) Publication that will specify the cryptographic algorithm. The proposed AES algorithm selected by NIST is Rijndael. See the section on cryptography for more details on this subject.

Languages like Java and JavaScript have been developed to embed programs into HyperText Markup Language (HTML) documents (pages). Java applets, which are designed to be downloaded from the web and run directly by the Java virtual machine within a browser, are also increasingly being included in web pages to provide more sophisticated animation and other desirable features. Downloading and executing code from anywhere on the Internet brings security problems along with it. That is, the host computer is open to a variety of attacks, ranging from attacks that simply monitor the environment to export information, to attacks that change the configuration or the behavior of the host (changing files, consuming resources), and finally to attacks that open a back door to let intruders get into the host.

Many security holes in web browsers have been discovered and brought to the attention of the public [DFW 96, DDK 98]. The web is constantly changing, and the browsers that are used for navigating the web are constantly being updated and improved. Most of the experiments described in the literature can no longer be reproduced using the latest versions of the web browsers. However, new attacks are constantly being discovered, and a systematic analysis of the features of both the Java and JavaScript languages and the web environment is needed to identify possible design weaknesses in order to avoid similar problems.

In the last five years, as network-based attacks have become more common and sophisticated, intrusion detection systems have shifted their focus from the hosts and their operating systems to the network itself. Intrusion detection began as a technique for detecting masqueraders and misfeasors in standalone systems [And 80, Den 87], but in the last few years the focus of intrusion detection has moved towards networks [MHL 94, VK 99]. In Network-oriented Intrusion Detection Systems (NIDSs) the source of events (and the object) for the analysis is a distributed system composed of many hosts and network links. The goal of NIDSs is to detect attacks that involve the network and may span different hosts. Network-based intrusion detection is challenging because network auditing produces large amounts of data, and different events related to a single intrusion may be visible in different places on the network.

NCSC's Trusted Network Interpretation (TNI) extended the TCSEC to deal with networks [NCS 87]. The TNI introduces two ways of viewing a secure network. The first view is the *Interconnected Accredited AIS View*. The assumption is that it is not practical to accredit the total system using the TCSEC, and the approach of accrediting individual parts for specific operational sensitivity ranges is adopted. When using this approach two accredited AIS devices may communicate at a range no greater than the intersection of their respective ranges. The second view is the *Single Trusted System View*. A common level of trust is exhibited throughout a single trusted system. "It is accredited as a single entity by a single accrediting authority." The single trusted system implements a reference monitor and has a trusted computing base (referred to as the Network Trusted Computer Base or NTCB). When using this approach, the sum of the component security policies must be shown to enforce the overall network security policy. Therefore, the network design must completely and unambiguously define the security functionality of components as well as between components. In addition, for accreditation the network architecture must demonstrate the linkage between the "connection-oriented abstraction" and its realization in the individual components. The problem is that the TNI is devoted almost exclusively to the single system view, while most secure networks are not built as a single system, but rather as interconnected components. In the coming decade more attention will be given to the interconnected trusted component approach.

Privacy Concerns

As computers and particularly databases are being used by most large corporations and government agencies to store and retrieve information on individuals, there is an increasing need to protect that data. Because these systems contain sensitive information on individuals, such as medical and financial records, if one of these systems is compromised, an individual may be harmed even though it is not that individual's computer [NRC 97]. As a result, there is likely to be an increased concern for the privacy rights of individuals in the next decade.

In the coming decade one can also expect to see new legislation for the handling of data sensitive to an individual's privacy. The trend of computer security, which has moved from exclusively defense oriented into a defense and commercial domain, is likely to concentrate more on privacy issues than has been the case in the past.

Trusted Systems

As computers are being embedded in more safety-critical systems, such as nuclear reactors, power control plants, aircraft auto-pilots and collision avoidance systems, and medical instruments, there is a need to expand the purview of computer security to a broader "trusted systems" view. This broadening is already taking place and the trend will continue. The new view of trusted systems will be systems that are secure, safe, and dependable. See the section on safety-critical systems for more details.

Another concern with the use of computers in safety-critical systems is that terrorists may start attacking these systems. The AT&T long distance network failure in 1989 is evidence of the vulnerability of some of these supposedly dependable systems.

Computer Security Education

Currently there is a shortage of individuals who are trained in designing, building, and managing secure systems. The National Research Council's System Security Study Committee also identified the lack of university-based research in their 1991 report [NRC 91]. Although activity in this area has increased in the last few years, more is still needed, and the need will increase over the next decade.

One suggestion is to introduce security awareness training as part of the initial introduction to computers. Computer security should be introduced in the elementary schools where students are currently learning how to use computers. The security awareness at this level is likely to be more successful if introduced as an awareness of the possibility of accidental loss of data or programs. Students should realize that they may lose their data due to a hardware failure or another student inadvertently destroying their data. The students should also be made aware of the ethics of the situation. They should know that destroying someone else's data is not a joke, but should be taken seriously. In the more advanced grades (e.g., middle school and high school) computer modules should include the possibility of malicious destruction of data. The students again should be made aware of the possibility and should know that doing this is unethical. They should also be made aware that copying someone else's programs, etc. without their permission is like plagiarizing their work or like cheating on a test.

At the college level modules to be included in the undergraduate courses already exist, although they may not be taught due to the lack of qualified instructors. In June 1987, the NCSC had a workshop to create modules for this purpose. The result was the design of computer security modules for the following courses: basic concepts course, operating systems, database, software engineering (formal specification and verification and risk analysis), and networks. These modules are available as Institute of Defense Analysis Reports [IDA 87]. The SEI also has a document on computer security modules. These modules should be included in all undergraduate computer science curriculum.

Finally, as a near term solution, government and industry need to invest in training their personnel in computer security. As these organizations are made aware of the vulnerabilities of their systems they will be more willing to educate project members in the techniques needed for building, maintaining and managing secure systems. Eventually funding for this training will begin to show up as line items in project budgets.

References

[Amo 99] Amoroso, E.G., *Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Trace Back, Traps, and Response*, Intrusion.net, 1999.

[And 72] Anderson, J.P., et al., "Computer Security Technology Planning Study," Deputy for Command and Management Systems, HQ Electronic Systems Division (AFSC), ESD-TR-73-51 Vol. 1, 1972.

[And 80] Anderson, J.P., "Computer Security Threat Monitoring and Surveillance," James P. Anderson Co., Fort Washington, PA, April 1980.

[Bac 00] Bace, R.G., *Intrusion Detection*, Macmillan Technical Publishing, 2000.

[BM 01] Bace, R. and P. Mell, "NIST Special Publication on Intrusion Detection Systems," National Institute of Standards and Technology, draft document, February 2001.

[Bev 89] Bevier, W. R., "Kit: A Study in Operating System Verification," *IEEE Transactions on Software Engineering,* Vol. 15, No. 11, November, 1989.

[CC 98] *Common Criteria for Information Technology Security Evaluation*, version 2.0, CCIB-98-026, Common Criteria Implementation Board, May 1998.

[CC 99] *Common Criteria for Information Technology Security Evaluation*, version 2.1, Common Criteria Implementation Board, December 1999.

[CSSC 93] *The Canadian Trusted Computer Products Evaluation Criteria,* Version 3.0e, Canadian System Security Centre, January 1993.

[COV 90] Minutes of the First Workshop on Covert Channel Analysis, *IEEE Cipher*, Los Angeles, CA, July 1990.

[DFW 96] Dean, D., E.W. Felten and D.S. Wallach, "Java Security: From HotJava to Netscape and Beyond," *Proceedings of the IEEE Symposium on Security and Privacy,* Oakland, CA, May 1996.

[Den 76] Denning, D.E., "A Lattice Model of Secure Information Flow," *Communications of the ACM*, Vol. 19, No. 5, pp. 236-243, May 1976.

[Den 82] Denning, D.E., *Cryptography and Data Security,* Addison Wesley, Reading, Massachusetts, 1982.

[Den 87] Denning, D.E., "An Intrusion Detection Model," *IEEE Transactions on Software Engineering,* Vol. 13, No. 2, February 1987.

[DDK 98] De Paoli, F., A. dos Santos and R.A. Kemmerer, "Web Browsers and Security," in Mobile Agents and Security, G. Vigna, Ed., *Lecture Notes in Computer Science*, Vol. 1419, pp. 235-256, Springer-Verlag, June 1998.

[DoD 83] *Department of Defense Trusted Computer System Evaluation Criteria,* Computer Security Center Standard, CSC-STD-001-83, 1983, in December 1983 released as a DoD standard, DOD 5200.28-STD.

[DTI 89] DTI Commercial Computer Security Centre, "Overview of Documentation," Vol. 1, version 3.0, February 1989.

[FS 90] Farmer, D. and E.H. Spafford, "The COPS Security Checker System," *Proceedings of the Summer 1990 Usenix Conference*, Anaheim, CA, June 1990.

[FC 92] *Federal Criteria for Information Technology Security*, Draft, Vol. 1, National Institute of Standards and Technology and the National Security Agency, December 1992.

[FHSL 96] Forrest, S., S.A. Hofmeyr, A. Somayaji, and T.A. Longstaff, "A Sense of Self for Unix Processes," *Proceedings 1986 Symposium on Security and Privacy*, Oakland, CA, IEEE, New York, pp. 120-128, May 1996.

[Fra 83] Fraim, L., "Scomp: A Solution to the Multilevel Security Problem, *Computer,* Vol. 16, No. 7, pp. 26-34, July 1983.

[GLP 79] Gold, B.D., R.R. Linde, R.J. Peeler, M. Schaefer, J.F. Scheid, and P.D. Ward, "A Security Retrofit of VM/370," *Proceedings of the National Computer Conference,* Vol. 48, AFIPS Press Montvale, N.J., 1979.

[GM 82] Goguen, J. and J. Meseguer, "Security Policies and Security Models," *Proceedings 1982 Symposium on Security and Privacy*, Oakland, CA, IEEE, New York, pp. 11-20, April 1982.

[GHM 78] Guttag, J., E. Horowitz, and D. Musser, "Abstract Data Types and Software Validation," *Communications of the ACM,* Vol. 21, No. 12, pp. 1048-1064, December 1978.

[HKM 87] Haigh, J.T., R.A. Kemmerer, J. McHugh, and W.D. Young, "An Experience Using Two Covert Channel Analysis Techniques on a Real System Design," *IEEE Transactions on Software Engineering,* Vol. SE-13, No. 2, February 1987.

[HGB 80] Hebbard, B., et al., "A Penetration Analysis of the Michigan Terminal System," *ACM Operating Systems Review,* Vol. 14, No. 1, January 1980.

[Hoa 72] Hoare, C.A.R., "Proof of Correctness of Data Representations," *Acta Informatica,* Vol. 1, pp. 271-281, 1972.

[Hu 91] Hu, W.M., "Reducing Timing Channels with Fuzzy Time," *Proceedings of the 1991 Symposium on Research in Security and Privacy,* Oakland, California, May 1991.

[IDA 87] Institute for Defense Analysis memorandum reports, M-379 through M-384, IDA, Alexandria, VA, October 1987.

[IKP 95] Ilgun, K., R.A. Kemmerer and P.A. Porras, "State Transition Analysis: A Rule-Based Intrusion Detection System," *IEEE Transactions on Software Engineering,* Vol. 21, No. 3, March 1995.

[ITS 90] *Information Technology Security Evaluation Criteria (ITSEC)*, Netherlands National Comsec Agency, Hague, The Netherlands, May 1990.

[JV 94] Javitz, H.S. and A. Valdes, "The NIDES Statistical Component Description and Justification," Technical Report, SRI International, Menlo Park, CA, March 1994.

[KW 91] Karger, P.A. and J.C. Wray, "Storage Channels in Disk Arm Optimization," *Proceedings of the 1991 Symposium on Research in Security and Privacy,* Oakland, California, May 1991.

[Kem 83] Kemmerer, R.A., "Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channels," *ACM Transactions on Computer Systems,* Vol. 1, No. 3, August 1983.

[Kem 90] Kemmerer, R.A., "Integrating Formal Methods into the Development Process," *IEEE Software,* pp. 37-50, September, 1990.

[KP 91] Kemmerer, R.A. and P.A. Porras, "Covert Flow Trees: A Visual Approach to Analyzing Covert Storage Channels," *IEEE Transactions on Software Engineering,* Vol. 17, No. 11, pp. 1166-1185, November 1991.

[KRL 97] "Execution Monitoring of Security-Critical Programs in Distributed Systems: A Specification-based Approach," *Proceedings of the 1997 IEEE Symposium on Security and Privacy,* pp. 175-187, May 1997.

[Lam 73] Lampson, B.W., "A Note on the Confinement Problem," *Communications of the ACM,* Vol. 16, pp. 613-615, October 1973.

[Lin 75] Linde, R.R., "Operating System Penetration," *Proceedings of National Computer Conference,* Vol 44, AFIPS Press, Montvale, N.J., 1975.

[LP 99] Lindqvist, U., and P.A. Porras, "Detecting Computer and Network Misuse with the Production-Based Expert System Toolset (P-BEST)," *Proceedings of the IEEE Symposium on Security and Privacy,* Oakland, CA., May 1999.

[Lip 75] Lipner, S.B., "A Comment on the Confinement Problem," *Proceedings of the Fifth Symposium on Operating Systems Principles,* The University of Texas at Austin, November 1975.

[Lun 88] Lunt, T.F., "Automated Audit Trail Analysis and Intrusion Detection: A Survey," *Proceedings of the llth National Computer Security Conference,* Baltimore, MD, Oct. 1988.

[MD 79] McCauley, E. and P. Drognowski, "KSOS: The Design of a Secure Operating System," *Proceedings of the National Computer Conference,* AFIPS Press, June 1979.

[MSW 90] McAuliffe, N.J., L.J. Schaefer, D.M. Wolcott, T.K. Haley, N.L. Kelem and B.S. Hubbard, "Is Your Computer Being Misused? A Survey of Current Intrusion Detection System Technology," *Proceeding of the Sixth Computer Security Applications Conference,* Dec. 1990.

[Mil 76] Millen, J.K., "Security Kernel Validation in Practice," *Communications of the ACM,* Vol. 19, pp. 243-250, May 1976.

[MT 79] Morris, R. and K. Thompson, "Password Security: A Case History," *Communications of the ACM,* Vol. 22, No. 11, November 1979.

[MHL 94] Mukherjee, B., L.T. Heberlein and K.N. Levitt, "Network Intrusion Detection," *IEEE Network,* pp. 26-41, May/June 1994.

[NCS 87] National Computer Security Center, *Trusted Network Interpretation of the Trusted Systems Evaluation Criteria,* NCSC-TG-005, Version 1, July 1987.

[NRC 89] National Research Council, *Growing Vulnerabilty of the Public Switched Networks: Implications for National Security Emergency Preparedness,* National Academy Press, Washington, DC, 1989.

[NRC 91] National Research Council, *Computers at Risk: Safe Computing in the Information Age*, National Academy Press, Washington, DC, 1991.

[NRC 97] National Research Council, *For the Record: Protecting Electronic Health Information,* National Academy Press, Washington, DC, 1997.

[Neu 90] Neumann, P.G., "A Comparative Anatomy of Computer System/Network Anomaly Detection Systems," assembled by Peter G. Neumann, CSL, SRI BN-168, Menlo Park, CA, May 1990.

[NNM 01] Northcutt, S., J. Novak, and D. McLachlan, *Network Intrusion Detection: an Analyst's Handbook*, New Riders, Indianapolis, IN, 2001.

[NSS 00] NSS Group, "Intrusion Detection and Vulnerability Assessment," Group Test (Edition 1), an NSS Group Report, Oakwood House, Wennington, Cambridgeshire, PE28 2LX , UK, 2000.

[Pop 74] Popek, G.J., "Protection Structures," *Computer,* June 1974.

[SS 75] Saltzer, J.H. and M.D. Schroeder, "The Protection of Information in Computer Systems," *Proceedings of the IEEE,* Vol. 63, No. 9, pp. 1278-1308, September 1975.

[SCS 77] Schroeder, M.D., D. Clark, J.H. Saltzer, "The MULTICS Kernel Design Project," *Proceedings of the 6th Symposium on Operating Systems Principles,"* 1977.

[SU 99] Sekar, R. and P. Uppuluri, "Synthesizing Fast Intrusion Detection/Prevention Systems from High-Level Specifications," *Proceedings of the Usenix Security Symposium},* 1999

[Sil 83] Silverman, J.M., ""Reflections on the Verification of the Security of an Operating System Kernel," *Communications of the ACM*, 1983.

[Sun 91] Sun Microsystems, Inc., "Installing, Administering, and Using the Basic Security Module," 2550 Garcia Ave., Mountain View, CA 94043, December 1991.

[VL 89] Vaccaro, H.S. and G.E. Liepins, "Detection of Anomalous Computer Session Activity," *Proceeding of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, May 1989.

[VK 99] Vigna, G. and R.A. Kemmerer, "NetSTAT: A Network-based Intrusion Detection System," *Journal of Computer Security*, Vol. 7, No. 1, pp. 37-71, IOS Press, 1999

[WKP 80] Walker, B.W., R.A. Kemmerer, and G.J. Popek, "Specification and Verification of the UCLA Unix Security Kernel," *Communications of the ACM,* Vol. 23, pp. 118-131, February 1980.

[War 70] Ware, W.H., "Security Controls for Computer Systems," The Rand Corporation, classified document, February 1970, reissued as an unclassified document R-609-1, October 1979.

[ZSI 89] Zentralstelle fur Sicherheit in der Informationstecknik, *IT-Security Criteria: Criteria for the Evaluation of Trustworthiness of Information Technology (IT) Systems*, Koln, West Germany, 1989.