Abstract

This chapter addresses systems and software security in computing environments over the past thirty years. It is partially a historical treatment of the subject which outlines initial efforts to bound the security problem beginning early 1970 through today’s state of security engineering practice. It includes an introduction to the topic, definitions and explanations necessary for background, design principles that were established by researchers and practiced today by security engineers. Government programs are described as they were initiated over time and their applicability to today’s security engineering practice is discussed. Important law and Executive decisions are included which have had a direct impact on the state of computer security today. Lessons learned by practicing security engineers are included as a part of concluding remarks. Much progress has been accomplished in research and practice, yet systems today appear as vulnerable or perhaps more vulnerable than they were in the past. This should not be necessarily interrupted as a lack of progress – but as an indication of the complexity of the problem being addressed and the changing nature of systems and networks needing the protection.
1. Introduction

At no time in our past history has there been more emphasis placed on the means and methods to secure systems and software than at the present time. Although much of this focus came to public attention following the tragic events of September 11, 2001, efforts to bring trust and security to software and systems have actually been underway for approximately the past 30 years or so. This chapter addresses this progress, beginning with initial efforts by the federal government and ending with thoughts on what the future holds. The chapter is written for those who are not specialists in this area, yet wish to understand the field at the generalist level. Background and definitional material is provided in section 2 to facilitate concepts presented later and references are periodically given for those that desire to pursue any specific topic to greater depth. When web links are available and likely to remain present for the foreseeable future, they too are included for the reader’s benefit.

The organization of this chapter is essentially chronological. A section discussing terminology and concepts is presented following the introduction to assist the reader with the context and content that follows. A section on historical perspectives follows which documents the many efforts in research and policy to bring trust, confidence, and correctness to our systems and software. The historical presentation contains comment and observations made by the author that often represents personal opinion and experience over the years. Sections then follow to discuss the state of current practice, security engineering capabilities, examples of what problems remain unsolved, and what the future may hold in this important area. Along the way, the reader may encounter recommendations and need for additional research effort. This chapter is not offered as a
comprehensive treatment of the subject of computer security (aka, information assurance). Many good textbooks have been published on this subject yet none of them would likely claim to be comprehensive. The problem of providing a secure solution for a specific software system or network is simply too difficult and too complex to be reduced to a single text or to a set of practices that will guarantee success. While there are commonly accepted best practices and some fundamental science involved, most would agree that securing systems and networks is today primarily a function of experienced, skilled, and knowledgeable systems administrators who constantly monitor and improve the protection mechanisms associated with that which is being protected.

2. Concepts of Information Assurance

2.1 Overview.

Information assurance or IA can mean something different to various individuals – depending on their position, interest, business, employer, and many other factors. To some, the focus is on a network, to others the interest may be on a particular software system or may involve a complete systems engineering effort (which should include hardware, software, people, processes, and procedure). In truth, security in any automation environment (networks, software, systems of systems, etc.) must take a holistic approach and include a complete systems engineering view – to include people and process as well as hardware and software. An attack or threat of any kind (intentional or unintentional) has a high likelihood of occurring at the point of the least resistance (or the weakest link). A comprehensive defense means that each point of attack or risk must be identified, assessed for threat, and some sufficient defensive
measure taken or planned. When we say holistic treatment or comprehensive approach, this normally means that the security engineer must consider hardware security, software security, policy, procedures, personnel employed, electronic emission security, and physical protection when constructing a protection strategy. Failure to properly address security in any one of these areas can and will introduce a point of vulnerability.

The process just described is in no way trivial and forms the essence of the information assurance discussion that follows and the research that has occurred over the past three decades. First, we need to make a distinction between the study of penetrating systems and that of defending systems. These two areas are sometimes referred to as offensive operations and defensive operations. On the offensive side, the penetration of systems can be a legal or illegal activity. Illegal activities may include authorized individuals inside the organization (we refer to this as an insider threat) who are misusing or abusing the computing assets in some manner that results in a violation of law or policy. This may be intentional or unintentional on the part of the employee. Examples of such actions could be installation of an unauthorized modem connection in a corporate network that then provides an unprotected entry point into otherwise protected assets (unintentional side effect), deletion of important and sensitive data by a disgruntled employee (intentional action), or use of corporate hardware and software to establish and run an adult web site (intentional action). In each case, damage to the organization occurs and policy (or perhaps law) is likely violated. Illegal activity can often occur from outside the organization (we refer to this as an outsider or external threat) by an unauthorized user. This again can be intentional or unintentional, although the preponderance of such activity is assumed intentional. Examples of such attacks might
include overwhelming an organization’s networks by sending a very large amount of spurious traffic to a single entry point and causing a system failure (or at least an availability issue), discovering a flaw in the operating system that allows the outsider to bypass protection and gain entry to a system (a hacker threat), downloading or receiving harmful (malicious) code or email that damages the system (unintentional outsider), or providing easy access to an unauthorized outsider which results in their ability to penetrate system defenses (e.g., providing one’s password and user identification by failing to properly protect it).

While illegal activities might be more intuitive to the reader, legal (from the U.S. point of view) penetrations are also a subject of interest to readers. A common method of testing system defenses is known as “penetration testing” – a sanctioned activity to attempt (in a controlled fashion) to gain unauthorized access. This form of testing can be accomplished by insiders (employees of the organization involved in testing system security) or outsiders (third party testing). In either case, the penetration is done on behalf of the organization with the intention of discovering weak defenses so that they can be shored up. It has been argued that this form of testing is one of the more useful measures of a system’s overall protection. Additionally, there is another form of penetration that is not often considered on first thought – that of information warfare (or information operations, network centric warfare, or other similar terms). Government sponsored technical staffs generally accomplish this form of penetration activity as part of its intelligence gathering activities or in preparation for military action (declared or undeclared) against another government. This may include mapping networks to discover topologies and weak points, covert insertion of malicious code, denial of service
attacks, or traffic analysis. We will not further address this form of legal penetration testing but wish to include mention of it here for the sake of completeness.

On the defensive side of information assurance, the security engineer tries to create what is referred to as a defensive perimeter (also known as a security perimeter) around the object of protection (e.g., the system, a network, a single host, a physical facility). The objective is to create enough penetration difficulty for the attacker so that the level of effort to penetrate exceeds the value gained if the penetration is successful. This is the same concept one uses when securing a home against possible invaders – that is, to make the level of effort necessary to gain entry more difficult than the attacker is willing to exert. Just as no home is one hundred percent secure, no useful computer system can guarantee total security. Because we cannot guarantee total security and because there is always risk present, we tend to use the term assurance to mean strength of protection. High assurance means a very strong system in terms of the security protection it offers and low assurance means very little security protection. A better characterization might be to think of assurance as trust in a system’s ability to provide protection. Trust can come from many sources – experience, examination of the code, testing, certification by experts, and others. Hence, this chapter is concerned with information assurance (a term that indicates a property having various scalable values) more so than computer security (an objective that is either present or not).

The degree of assurance that we arrive at for a system (through whatever means we employ) is not a static value. That is, a high assurance system yesterday may be reduced to a low assurance system tomorrow though many means. This might occur because a previously undiscovered vulnerability in a specific operating system is
announced, an update to the software configuration is installed with flawed code, a lapse in security procedures occurs due to a change in key personnel, or a firewall is accidentally misconfigured by a systems administrator. Many other examples could be provided here, but the point is that past performance in terms of trust says very little about future performance characteristics. It is incumbent on the systems and/or security engineer to continuously update the security perimeter and to check on its effectiveness. There is simply no guarantee of impenetrability and today, much of our reliance is on the individual skill set of those that administer the system and its protection. The advantage always lies with the attacker of a system in that, with patience, the attacker must only find a single unguarded entry point into the system while the defender must block them all. Even when penetration is prevented, the majority of systems are certainly subject to being taken out of service though a denial of service attack, which may overwhelm its processing capacity or network resources to the point of failure.

2.2 Background Concepts.

Initially, security for computing systems was thought to be primarily an issue of access control to data so that users could be assured that others having access to the computer system could not obtain access to data in the system that they did not have permission to see. In other words, early attempts to protect systems worked within a confidentiality paradigm. Section 3 will address more of the rationale for this early view. As the professional and technical communities learned more about the security problem and the needs of specific organizations, other definitions began to surface that were not confidentiality issues. In fact, before a security engineer can accomplish any work, it is important that a firm understanding of the organizational definition of security be
achieved and agreed to. This is often referred to as the security “policy” for an organization. Once a firm understanding of the policy is obtained, the security engineer (working with the organization) may develop a “model” of proper system behavior which complies with the policy and which can serve as a system specification. Common security concerns today are generally one of or a combination of the following characteristics:

- **Secrecy or Confidentiality**: A guarantee that assets of a protected system are accessible only by authorized parties.

- **Integrity**: Data assets can only be modified by authorized parties and only in authorized ways. Related issues include the preciseness of the data, consistency of data, and accuracy of the data.

- **Availability**: Computing assets (e.g., data, processing power, network bandwidth) are accessible to authorized parties when needed. Note that the absence of this characteristic is known as a *denial of service*.

- **Accountability**: The ability to establish correspondence between an action on the system and the responsibility for the action. This characteristic also includes a more specific area called *non-repudiation*, which is the establishment of responsibility for a computing action that cannot be denied.

A primary object of concern when developing a protection strategy is the data maintained by the computing system. Data must be in one of three electronic states – processing (active manipulation), storage (passive or at rest), or in transmission. The security engineer’s objective then becomes to preserve the security characteristics of interest to the organization (i.e., confidentiality, availability, integrity, accountability) across all
possible states of processing – a very difficult problem and one which involves significant analysis, testing, the weaving together of several disparate products (also known as mechanisms) into a protection perimeter, and then monitoring its effectiveness over time.

Often information security is thought to be synonymous with encryption. While encryption is indeed a strong tool to use in developing a protection scheme, it is not a complete solution. When one considers how to protect data in transmission, encryption is the obvious solution, but it does not adequately or completely address protection needs while data is being processed or stored. Strong encryption is useful to keep data confidential while transiting a network for example. It is also a means to guarantee integrity of the data in that unauthorized modification of encrypted packets is likely detectable when the data is deciphered at the receiving end. Because of these characteristics, encryption has been the tool of choice in not only protecting data while moving it over a network, but also in specific protocols that result in the ability to digitally sign an electronic document in such a manner that the signature cannot be repudiated; to exchange encryption keys securely, and to carry out secure dialogue in a client server environment. Encryption is more of a commodity today to those that must employ it (with the exception being the intelligence community who create their own). It comes preinstalled in web browsers, is available in many commercial off-the-shelf products, and is available as freeware from web sites. Encryption strength is always a subject of concern and the length of time needed for a dedicated adversary to break it changes over time as computing technology improves. Web browser security offers sufficient security today for web transactions, but would not be sufficient to protect
national secrets. The complete study of encryption and its progress through the years is beyond the scope of this chapter. It is treated here as one of many mechanisms that the security engineer must consider using in providing a total solution.

When taking a necessary holistic approach to the security problem, there are many areas of concern that encryption will not solve. There are many examples of such areas and the following are provided as a representative sampling.

- Physical controls. A good security perimeter requires solid physical controls so that an attacker does not have access to the equipment, the network wiring, employee workspace, files, or other sensitive items. A lack of such controls can result in the theft of equipment which contains sensitive data, a social engineering attack where the attacker uses human error or a lack of attention to policy and procedure to discover sensitive information or to gain access, the installation of a network recording device (called a sniffer) to record sensitive packets, or the installation of malicious software (e.g., a trap door or Trojan horse which can be used later by the attacker to gain system access). Good physical controls are necessary to protect against the outsider threat.

- Policy. Written policy and procedures that are made known to all authorized users of the system and enforced are paramount to overall system security. Studies in the past have shown that a large number of reported security incidents are unintentional, insider actions – that is, mistakes by an otherwise well-intentioned employee. Written policy, which is known and understood by employees, addresses this vulnerability. Examples of such policy might include disallowing an employee to install a dialup modem on their office computer because such a device may introduce
an unprotected entry point into the corporate network; prohibiting employees from downloading Active X controls (a form of what is known as mobile code) from a web server (accomplished by a simple browser setting) because such action can introduce malicious code into the corporate system; or, establishing a policy requiring frequent changing of passwords that are at least 8 alphanumeric characters long as a preventative measure against password discovery (e.g., cracking attacks). Policy is an excellent non-technical risk mitigation strategy – but only if employees understand the policy and the reason for it.

- **Software controls.** Any useful computing system contains a vast quantity of software integrated into a suite of services and functionality for the user. Software can contain intentional or unintentional hidden functionality that might be exploited by an attacker. Acquiring software with high assurance mitigates risk associated with software that might be of lesser assurance. Using an operating system that has been certified as having high assurance by a government agency (e.g., the National Institute of Standards and Technology) is preferred over using one that has not. Acquiring application software from a responsible and reputable source is preferred over downloading such software from an unknown author who may have posted it free of charge on a web site. This area of concern also includes protection against malicious code attacks such as computer viruses, worms, or Trojan horses. Charles (Chuck) Pfleeger addresses this topic nicely in Chapter 5 of his book, Security in Computing.³

- **Inference or aggregation.** In the modern world of Internet connectivity and web servers, organizations must exercise caution with respect to the amount of
information that is placed in various publicly available areas that might be independently accessed and combined together by an attacker to gain confidential or sensitive insights that should not have been publicly disclosed. This is referred to as the problem of aggregation. An example might include a corporate web site that on a general news page carries an announcement that there are confidential discussions underway with a competing company that may lead to a merger. A link from the main page to the human resources page may contain new information on a plan to move to a new employee benefit program that is generally known to be a benefits provider for competitor X. A third link may appear on the corporate web page housing “other links of interest” to competitor X’s web site. Each piece of information by itself is not sensitive – but combined it begins to leak sensitive information to the detriment of the organization. Inference is a close relative of aggregation and occurs when non-sensitive information is obtained from a computer system and mathematically manipulated to discover sensitive information. Inference is generally associated with database queries, but need not be strictly isolated to that area. An example might include obtaining the total of all executive salaries in a small organization (non-sensitive aggregate data) and subtracting from that the total of all male executive salaries in the same organization (non-sensitive aggregate data) to arrive at the salary for the only female executive in the organization (sensitive, specific data). Defenses again inference and aggregation attacks are difficult to achieve since in the majority of cases, the attacker uses information obtained outside the organization and combines it with information obtained freely and legally from the organization resulting in disclosures that should not have been allowed.
The examples above were provided to demonstrate the wide area of concern that a practicing security engineer must be concerned with and to show that a single solution set is not adequate for an organization. It is imperative that a suite of products, policy, procedure, and training be combined by a knowledgeable engineer and constantly monitored over time if risk of attack is to be reduced to an acceptable level. This is often more art than science today.

3. A Historical Perspective

3.1 Introduction.

Protection of information while in transit has far deeper roots than does computing security. The use of cryptography, our most useful tool in this regard, can be traced back more than 4000 years to ancient Egypt and in some reports, even earlier (for a good overview of historical notes, the reader is invited to review Ron Gove’s historical perspectives in the Information Security Management Handbook). The history of the origin of computing machines is somewhat more recent and can be traced back to the seventeenth century when gear driven computing machines were described and constructed. Credit for these advances is often given to notable early scientists such as Blaise Pascal of France, Gottfried Wilhelm Leibniz of Germany, and Charles Babbage of England. All of these machines were capable of following some algorithm (a precursor of computer programming). Even the historically famous automated weaving loom developed by Joseph Jacquard of France in the early 1800’s was an example of a programmable computing device used to control an industrial process. Security in these devices was not an architectural consideration and some would agree that even if security were a recognized need, it was provided for by the obscurity of the technology and the
high level of knowledge needed to understand the mechanics and mathematics associated with these devices. Similar thought processes occurred when electronics was applied to these mechanical structures and modern day computers began to evolve. During the early 1940’s, the truly first computers (as we know them today) were built. These included early prototype work at Bell Laboratories, Harvard University (the Mark I computer), Iowa State University (the Atanasoff-Berry machine), the University of Pennsylvania (ENIAC), and a code breaking machine developed in England for the intelligence community, known as the COLOSSUS. Other machines followed over the years, gradually moving from highly specialized and dedicated machines to general purpose computers that were cost effective for use in administering business or government, as well as useful in scientific exploration. Interestingly, the time between the introduction of the first prototype specialized computers and their widespread general use was only approximately 20 years. During this evolution of general purpose computing, engineering emphasis was placed on ease of use, speed, operating systems, programming languages, utility software, storage advances, and memory to name a few. Security, as a serious requirement, was not a high priority or even a major consideration beyond that needed for some basic auditing (for cost accounting purposes). Physical access controls were generally thought to be sufficient to address any other concerns in this area. During the decade 1960-1970, however, security and protection of information assets began to assert itself as a requirement in modern computing systems and one that could be difficult to resolve. The interest in security was being driven more by evolving technology at this time than by a user demand. Computing machines were becoming more affordable and as a result were proliferating quickly into society at large.
Additionally, advances such as multiprogramming, networking, disk storage, large and persistent memories, application layer programs, resource sharing, and databases were increasing the amount of shared space and data exposure. The serious study of computer security began toward the end of this decade (circa 1967) sponsored by the U.S. Department of Defense as a response to growing concern with resource sharing computers and the risk they posed to loss of National classified information. One of the earliest reports addressing this concern was a Defense Science Board report\(^5\) titled “Security Controls for Computer Systems” that was chaired by Dr. Willis H. Ware, then of the RAND Corporation. This paper is often cited as seminal work and the first that truly outlined the security problem in computing. It set in motion a series of events and responses by the Department of Defense (DOD) and the Federal government that resulted in international impact and advances through research that continue today. The historical overview that follows in this section is intended to provide the reader with a broad sense of past efforts and where they have led us over time to the present day. In the large, the problem remains unsolved but important progress in understanding the problem has been made.

The remainder of this section is an attempt to overview significant events beginning with the 1970 Defense Science Board (DSB) report and ending with perspectives on where we are today. Apologies are extended if in the opinion of the reader an important historical event is omitted in the remainder of this chapter. Admittedly, the events discussed here are based on this author’s opinion of their importance.

3.2 The Defense Science Board Report.
This report is generally considered the first major scientific work reviewing and documenting the computer security problem. It was commissioned in the summer of 1967 by the Department of Defense (the Assistant Secretary of Defense, Deputy Director for Administration, Evaluation, and Management) in response to growing concerns that computer systems were proliferating throughout the military which were then being used to process and store sensitive information. Both defense contractors and DOD technical staffs were pressing the issue of security, the need for appropriate policy, and safeguards.

The task eventually fell to the Advanced Research Projects Agency or ARPA (the forerunner of today’s Defense Advanced Research Projects Agency or DARPA) and a task force operating under the authority of the Defense Science Board was eventually formed with Dr. Willis Ware of the Rand Corporation as its chairperson. Two panels were organized to review the problem and to make recommendations – a technical panel and a policy panel. Membership on these panels was diverse and well chosen – many members later were recognized for their strong contributions to the problem of computer security and are known today for their seminal work in this area. A full list of the membership is contained in the report, which is publicly available on the web at http://seclab.cs.ucdavis.edu/projects/history (note: this report and other key early papers were collected and stored electronically for public review under a grant by the National Security Agency to the University of Maryland). The report was originally classified at the Confidential level by the DOD and later downgraded to unclassified and made publicly releasable.

The report clearly addressed the need for holistic security controls – a theme that still exists today. Technical measures as well as administrative policy and procedures
must all work together to address the security problem. They also characterized two important environments within which secure computing systems must operate. These environments still generally exist today, albeit with some modification as a result of technical advances over the years. They were identified as closed and open environments

where a closed environment was one that consisted only of trusted (or in the DOD vernacular, “cleared”) users, working at physically protected workstations, connected to a physically protected computer system by protected communication circuits (i.e., physical, cryptographic, and electronic protection). Such an environment offers opportunity for the construction of a high assurance system that can processes very sensitive information. This kind of closed system can reasonably be established in the commercial world also – for example, in the domain of banking. The other environment, and the more problematic, is known as open and is characterized as one in which there is a mixture of trusted and untrusted (or cleared/uncleared) users. The untrusted users use the system at unprotected workstations, connected to a central computing system by communicating over unprotected communication lines. The trusted users work from protected workstations and communicate over protected communication lines. Such an environment is far more difficult to establish assurance for and at the time of the DSB report, the authors believed that technology did not exist to fully address this problem

[note: There is good argument that this has not changed a lot in the ensuing 30 plus years]. Furthermore, in a memorandum to the Chairman of the Defense Science Board, Dr. Willis wrote, “Thus, the security problem of specific computer systems must, at this point in time, be solved on a case-by-case basis employing the best judgment of a team consisting of system programmers, technical hardware and communications specialists,
and security experts.” This same truth holds today in that security engineers employ best judgment in a specific environment against a specific set of threats.\textsuperscript{6} The conclusions reached by the task force in 1970, are reported verbatim below as taken from the memorandum written by Dr. Willis to the Chairman of the DSB and are annotated with this author’s comments and thoughts. The annotations are enclosed in brackets following each conclusion for ease of separation.

- Providing satisfactory security controls in a computer system is in itself a system design problem. A combination of hardware, software, communications, physical, personnel, and administrative-procedural safeguards is required for comprehensive security. In particular, software safeguards alone are not sufficient. [This conclusion holds today and refers to the need for a holistic approach by the security engineer. An attacker will penetrate the point of least resistance so a weakness in any of the areas identified in this conclusion will become a potential target. In general, most penetrations today are not technical.]

- Contemporary technology can provide a secure system acceptably resistant to external attack, accidental disclosures, internal subversion, and denial of use to legitimate users for a \textit{closed environment} (cleared users working with classified information at physically protected consoles connected to the system by protected communication circuits). [The key to this conclusion is the “acceptably resistant” phrase. This means that we can provide a sufficiently secure solution in most cases if we can assume trusted users and protected systems and networks. The same is true today].
- Contemporary technology cannot provide a security system in an open environment, which includes uncleared users working at physically unprotected consoles to the system by unprotected communications. [Recalling that an open environment means a mix of trusted and untrusted users coupled with protected and unprotected systems and networks, this conclusion is only partially true today. Research and advances in security engineering have allowed for a much greater degree of protection in open environments today than in 1970. While it is still unwise to place national secrets in such an environment, e-commerce and other business applications today operate in this environment quite comfortably with, in most cases, sufficient security.]

- It is unwise to incorporate classified or sensitive information in a system functioning in an open environment unless a significant risk of accidental disclosure can be accepted. [Again, most would agree that significant advances have been made with respect to this conclusion. While it is still unwise to mix highly sensitive information with public information in an open environment, some lower level sensitive information can be resident on the same system accessible by the public with adequate assurance of separation today.]

- Acceptable procedures and safeguards exist and can be implemented so that a system can function alternately in a closed environment and in an open environment. [This conclusion addresses a work around that the panel came up with, sometimes called periods processing. The procedure can still be used effectively today for a stand-alone system. It requires that the system be brought to a halt, all sensitive information is removed, and then the system is re-initialized for open processing].
Designers of secure systems are still on the steep part of the learning curve and much insight and operational experience with such systems is needed. [Most security engineers would agree that the learning curve is still steep and that operational experience and insights are still required].

Substantial improvement (e.g., cost, performance) in security-controlling systems can be expected if certain research areas can be successfully pursued. [This conclusion initiated significant government funding of trusted computing systems which continues today in various forms.]

Clearly, the authors of the DSB report recognized the lack of architectural guidelines for secure systems and the risk that the Federal government was taking in the use of resource sharing computers. The call for research in this area was not ignored and resulted in a research focus that continues today. Specifically, the DSB report called for research in the following areas:

- Facilitate progress toward handling the open environment. The development of encryption devices to function internally within the computer proper was seen as a strong need as was the development of special hardware configurations that could provide satisfactory security controls in an open environment.

- Improve the understanding of failure risks. This suggestion was designed to initiate a program of research leading to a better understanding of the processes and policy needed to certify and re-certify systems for sensitive processing. In today’s world, the federal government does have such processes in place, but there remains much room for improvement and the need for qualified certifiers remains.
- Improve the efficiency of security controlling systems. This suggestion reflected the need to develop new architectures for resource sharing computers that had, as a fundamental design principle, a security requirement. The members of the DSB study believed that with appropriate research focus new computer architectures could implement security controls more efficiently and correctly than present day systems did. They also recommended a parallel program of research to predict failure probabilities and failure modes in systems. The suggested focus on creating security architectures was taken to heart by the Department of Defense and major investments were made in this area throughout the 1970-1990 timeframe. In fact, this work continues in various forms even today.

- Solve a latent and not fully understood leakage point. In the context of the DSB report, resource-sharing computers were viewed as information repositories that were porous in nature with a tendency to “leak” protected information in a large variety of ways such that compromise of the information could occur. The report specifically mentioned leakage occurring from improper “erasing” of magnetic media – but within the entire report, many other leakage points were identified. This call for research was insightful for its time – particularly in the area of magnetic remanence where researchers were discovering the persistent nature of magnetic memory and that with highly specialized techniques, information could be retrieved even after having been erased. Additionally, this suggestion was directly related to the whole area known as object reuse which in the years following the DSB report became a required architectural component in trusted systems (note: object reuse will be
discussed later in this chapter, but involves the requirement to clear information from shared resources prior to allocating that resource for reuse by another subject).

The authors of the report also prophesized that although the Department of Defense had initiated the study on computer systems security controls, this subject would very soon transcend the DOD and become an area of interest to the entire federal government and industry. Although not specifically addressing this thought in detail, it proved entirely accurate as in the next few years confidentiality concerns in the DOD were joined by integrity and availability concerns from the private sector and from the federal government. As time passed sensitive but unclassified, privacy data, non-repudiation, and other such issues demonstrated that security in computer systems extended far beyond what was initially thought to be only a confidentiality problem.

This report has long been considered a seminal work in computer security and one that still deserves study today. It bounds the problem nicely for students and outlines the fundamental areas that are of concern to those that are charged with securing systems. Much of the work that is reported on in this chapter is a result of the DSB report and its recommendations.

3.3 The Reference Monitor

One of the DSB recommendations was for architectural change in computing systems such that security could be enhanced. An important response to that call was suggested by James P. Anderson\(^7\) in 1972, within a report prepared for the U.S. Air Force. In this report, Anderson outlined the need for strong access controls and recommended the use of hardware and software to implement a reference validation mechanism (later referred to as a reference monitor concept). Although there are fine
differences between the terms reference validation mechanism and reference monitor, they are used interchangeably in this chapter. The idea was elegant in notion and design and later became an important feature of operating system security and was adopted by the DOD as a fundamental architectural concept for trusted systems and was included in national level guidance and standards (specifically the DOD 5200.28-STD, the DOD Trusted Computer System Evaluation Criteria or “Orange Book”). It persists today in other trusted systems documents to include ISO Standard 15408 (The Common Criteria).

In the Anderson report, the computer system was modeled as a set of subjects (active entities in the system, e.g., processes) and a set of objects (passive entities in the system, e.g., file storage). Anderson suggested the use of a reference validation mechanism that would intercept every request by a subject for access to an object and validate that access based on a set of rules. These rules could, in effect, implement a security policy for the system – both discretionary (user/owner determined) and mandatory (organizationally determined) policy. In fact, the reference validation mechanism could be used to enforce a mandatory policy over a user’s desire to share information or provide access to another subject if the organizational policy would be violated as a result. Although a specific implementation of a reference monitor was not given in the report, three important properties that any implementation of it would have to have were provided. These are given below with brief explanation.

- The reference validation must be tamperproof. This requirement was one of protection against attack and modification. If the hardware, software, or firmware components were modified, then no guarantee of its proper action could be made for access control. In any trusted system that employs the reference monitor concept,
protection must be convincingly demonstrated to evaluators of that system. This characteristic is sometimes referred to as isolation.

- The reference validation mechanism must always be invoked. This characteristic is sometimes referred to as a no bypass capability or completeness. There must be no path in the system such that a subject can access an object without invoking the reference monitor. In practical systems, this rule is violated by privileged users such as system administrators or security officers, but each such exception increases the risk of a security failure.

- The reference validation mechanism must be small enough to be subject to analysis and tests so that its completeness can be assured. There is more requirement in this simple sentence than may first be observed by the reader. What is implied here is that the reference monitor must be correctly implemented and that correctness determination is enhanced if the mechanism is small enough to be comprehensible.

The word “assured” was used by Anderson, and in computer security study, that term (assurance) means “a degree of trust”. Trusted systems, for example, offer more “assurance” to the user that their information assets will be protected than do systems that are not trusted. Software from a reputable and known source generally offers more assurance that it has no malicious components than does software from unknown sources. The desire for the reference validation mechanism to be small stems from a desire to inspect (but formal inspection as well as informal inspection techniques) the software code and to use mathematical rigor to prove it correct for high assurance systems. This is known as verifiability.
An Air Force officer by the name of Roger Schell was actively involved in the study of computer system security vulnerabilities in the early 1970’s and is generally credited with first specifying the concept of a reference monitor in 1972, as a security kernel. The implementation of the reference monitor concept has become known as a *security kernel* since that time and over the past thirty years there have been many commercial attempts to implement one. Although existing reference monitors have primarily been implemented in software, the concept itself is not restricted to software implementations. During the late 1970’s several security kernel implementations were accomplished – primarily as research projects. MITRE is generally given credit for implementing the first on a DEC PDP-11/45 in 1975. For a full treatment of the advances in security kernels and the history behind them, the reader is invited to review Chapter 10 of Gasser’s excellent book titled *Building a Secure Computer System*.8

A closely related concept is the *trusted computing base* or TCB defined in the DOD Trusted Computing Systems Evaluation Criteria (TCSEC)9. The TCB was defined as the totality of protection mechanisms within a computer system – including hardware, firmware, and software – the combination of which is responsible for enforcing a security policy. The TCB concept is broader than the security kernel and in a system using the security kernel it becomes that kernel plus various trusted processes. Not all systems that are certified to process sensitive information use the reference monitor approach. The TCB description was offered as a matter of convenience so that protection components of a computer system could be described as a set of elements. The TCB is sometimes referred to as a *security perimeter* and includes all parts of the operating system that enforces a security policy. While it may seem better to use a reference monitor approach,
in reality it may not be possible with legacy systems. A reference monitor is an architectural decision that must be implemented during the initial construction of an operating system – not a retrofit added later. In cases where software systems not employing a reference monitor concept are used in sensitive environments, a description of the TCB may be sufficient to convince a certification authority that security policy enforcement is sufficient for the degree of assurance required by the system and its users.

3.4 More Architectural Principles

Corresponding with the increased research activity brought on by the 1970 DSB report and the emphasis being placed on the development of more trusted systems for use by the government, Jerome Saltzer and Michael Schroeder (in 1975)\textsuperscript{10} published eight fundamental principles for secure system design. These have been cited many times since and still today form an important basis for those constructing trusted systems or networks. They are repeated below with a brief synopsis taken from Saltzer and Schroeder with additional comments included in brackets.

- Economy of mechanism. It is important to keep the security design as small and as simple as possible. Security errors do not tend to manifest themselves in normal use and special examination techniques are required to find and eliminate them. By keeping the design small, examination and discovery of security errors is made less difficult. [This is also a common software engineering best practice. The principal can be difficult to implement in a security context since security functionality touches so many other areas in an overall system].

- Fail-safe defaults. Access permissions should be based on explicit permissions and not exclusion. Defaults should always be the more secure option. [In modern day
firewall practice, a similar practice is encouraged in that behavior not explicitly allowed is denied. Many operating systems today are shipped with defaults set to relatively open permission and the installer has to close these permissions explicitly. The fail-safe default principle would tell us that such systems should be shipped with all permissions turned off and only those that are to be allowed in operation should be explicitly turned on. There is debate as to whether or not this is a truly practical approach.]

- Complete mediation. Every access to every object must be validated and mediation cannot be bypassed. [This is a reinforcement of the reference monitor concept presented earlier].

- Open design. The design itself should not be the basis for secrecy nor should it be a secret at all. The more open is a design, the more likely it is that the community of reviewers will find flaws and repair them. [Salter and Schroeder might be considered early advocates of the open source movement. This argument is one that open source advocates espouse and claim that such software is more robust, efficient, and secure due to the inspection process that tends to occur with open source development. There has been other research that disputes this and claims made that there is ‘security in obscurity’.]

- Separation of privilege. This principle advocates not putting too much authority in a single mechanism and requires, for critical or sensitive applications, that two or more separate process cooperate to accomplish a function.

- Least privilege. It is important that all mechanisms run at the lowest privilege required to appropriately perform their intended function. If an application program
invokes a system utility, that utility should run, if at all possible, at the application program level or privilege and not at “root” level.

- Least-common mechanism. The idea behind this principle is to design system mechanisms such that they tend to operate on behalf of a single user and perform a single function. If we have a mechanism that operates on behalf of several users at the same time, we increase the risk of information compromise or having the mechanism itself used as a means to upgrade user privilege.

- Psychological acceptability. The security features used in a system must meet with user acceptance or they will be ignored or bypassed. Human factors engineering is important.

The Saltzer and Schroeder principles are not complete and there are other good best practices found in the literature, but their principles are the most widely cited. Inherent in the message they tried to communicate in their paper is another principle that has been widely regarded as key – build security into a system from the very beginning. Security must be a fundamental design objective and not an afterthought. One of the major difficulties faced today by practicing security engineers is to take an existing system or network that was not constructed with security as a design constraint and to then secure it. Adding security to a completed system is not effective in the majority of cases and can lead to what is known as a penetrate and patch philosophy – that is, various mechanisms are added to counter each penetration of the system.

3.5 A Government Infrastructure is Built

During the late 1970’s and early to mid 1980’s, the Federal Government began to build an infrastructure and publish standards and guidance in the area of computer
security and computer controls. This activity was partially in response to the DSB report discussed in 3.2 above (within the Department of Defense) and partially in response to US Congressional legislation – the Brooks Act of 1965. As members of the DSB had predicted, concerns with security issues were not unique to the DOD and would most certainly involve other government activities and industry. The Brooks Act specifically named the National Bureau of Standards or NBS (later renamed the National Institute of Standards and Technology or NIST) as the federal agency responsible for the creation and promulgation of federal computer standards, implementation, and research. This legislation officially marked the beginning of a long partnership between the National Security Agency (representing DOD) and the federal government in the area of computer security – a cooperative arrangement that continues today.

In response to the Brooks Act, NBS initiated several studies to define the security problem and to organize their own internal effort to begin standards production. In 1977, NBS began an important series of workshops¹¹ (by invitation only) to investigate the needs for audit and evaluation in computer systems. The goal of the workshops was to determine “What authoritative ways exist, or should exist, to decide whether a particular computer system is ‘secure enough’ for a particular intended environment or operation, and if a given system is not ‘secure enough’ what measures could or should be taken to make it so.” Nearly sixty attendees participated from the federal government and supporting contractors and their work resulted in a series of NBS publications - but more importantly the workshop formalized for the federal government, conclusions very similar to those already reached by the DSB in their report a few years earlier, that the provision of security in a computer system involved much more than just technical
Attention to policy, hardware and software mechanisms, and a way to measure the strength provided by the system (known as assurance) was needed. Secondly, the workshop concluded that no then present day operating system could be considered secure in terms of its ability to separate users of that system from data that they were not authorized to see. They noted that computer systems could of course, process highly sensitive data, but that specialized process and procedure had to be employed to protect the confidentiality of the data processed. Reliance on the operating system for such protection was not technologically possible. While this statement might not seem too insightful today, it represented a formal beginning to the quest for what today is known as multilevel security. A second NBS workshop was held the following year (1978) that resulted in a call for specific actions that needed to be taken. These included the need for a national computer security policy for information considered sensitive, yet not covered by existing policies (this later became known as “sensitive but unclassified” or SBU information); the need for the creation of a government process leading to formal evaluation and accreditation of computer systems; and, a formal technical means to measure and evaluate the overall security of a system (assurance evaluation). Embedded in this report was also a call for the government to maintain a list of “government approved” products that could be used for sensitive environments – essentially a list of evaluated products and whose assurance level was known and trusted. These recommendations were key events in the evolution of a government infrastructure that exists today to evaluate and recommend products. Also beginning in 1979, in response to the NBS workshops was the DOD Computer Security Initiative consisting of a series of workshops and resulting in a new mission assignment to the National Security Agency.
(NSA) in 1980 – that of promoting trusted information systems and products within the DOD (although their actual impact was far outside the DOD community). As an outcome of the NBS workshops and interest from the DOD, MITRE assumed the task of creating computer security evaluation criteria that could be used to measure assurance (trust) in a computer system. This project resulted in several documents and proposed criteria\textsuperscript{12} \textsuperscript{13} \textsuperscript{14} that formed the foundation for later DOD and national standards in this area.

3.5.1 Birth of the National Computer Security Center (NCSC)

In January of 1981, the DOD formally established and chartered what was then known as the DOD Computer Security Center (later renamed the National Computer Security Center or NCSC, in 1985). In the evolution of trusted systems, this was perhaps one of the most important events. Established originally by DOD Directive 5215.1 the Center was assigned the following goals.

- Encourage the development of trusted computer systems.
- Evaluation of the protection capability (strength) of systems.
- Provide technical support and advice to those involved in computer security R&D and conduct and sponsor research efforts.
- Develop and publish technical criteria to be used in the evaluation of computer systems and products.
- Apply these technical criteria to the evaluation of commercial computer systems and products.
- Develop tools to assist in building trusted systems.
- Publish computer security guidance.
Conduct computer security training.

One might notice at this point that both NIST and NSA were key players in the emerging government interest and standardization of computer security guidance and the efforts of one were consistent and supportive of the other. This strong relationship was forged over time and continues today. The initial MITRE efforts for NBS in the area of evaluation criteria became the foundation documents for the evaluation criteria published by the NSA as the DOD Trusted Computer System Evaluation Criteria in August 1983 and later re-released as DOD Standard 5200.28-STD in December 1985. This document became known as the Orange Book based on the color of its cover and, for the first time, established a method for evaluating and ranking trusted operating systems – satisfying at least part of the new mission assigned to NSA. While there will be no attempt to present the details of this landmark document here, the interested reader is invited review it in greater detail at http://www.radium.ncsc.mil/tpep/. The evaluation criteria itself consisted of seven hierarchical classes of assurance which were contained in four divisions (A, B, C, and D). Most divisions were further sub-divided into classes. Various security mechanisms were required at specific classes (e.g., mechanisms might include auditing of security relevant events or identification and authentication). At the higher levels of assurance, fewer mechanisms were required, but stronger assurance practices became necessary (e.g., security testing, covert channel analysis, and specification and verification procedures). A summary of the evaluation classes and their meaning as taken from the Trusted Computer System Evaluation Criteria (TCSEC) Appendix C, is provided in Table 1 and are presented in the order of increasing assurance.
<table>
<thead>
<tr>
<th>TCSEC CLASS</th>
<th>GENERAL DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class (D): Minimal Protection</td>
<td>This class is reserved for those systems that have been evaluated but that fail to meet the requirements for a higher evaluation class.</td>
</tr>
<tr>
<td>Class (C1): Discretionary Security Protection</td>
<td>The Trusted Computing Base (TCB) of a class (C1) system nominally satisfies the discretionary security requirements by providing separation of users and data. It incorporates some form of credible controls capable of enforcing access limitations on an individual basis, i.e., ostensibly suitable for allowing users to be able to protect project or private information and to keep other users from accidentally reading or destroying their data. The class (C1) environment is expected to be one of cooperating users processing data at the same level(s) of sensitivity.</td>
</tr>
<tr>
<td>Class (C2): Controlled Access Protection</td>
<td>Systems in this class enforce a more finely grained discretionary access control than (C1) systems, making users individually accountable for their actions through login procedures, auditing of security-relevant events, and resource isolation.</td>
</tr>
<tr>
<td>Class (B1): Labeled Security Protection</td>
<td>Class (B1) systems require all the features required for class (C2). In addition, an informal statement of the security policy model, data labeling, and mandatory access control over named subjects and objects must be present. The capability must exist for accurately labeling exported information. Any flaws identified by testing must be removed.</td>
</tr>
<tr>
<td>Class (B2): Structured Protection</td>
<td>In class (B2) systems, the TCB is based on a clearly defined and documented formal security policy model that requires the discretionary and mandatory access control enforcement found in class (B1) systems be extended to all subjects and objects in the ADP system. In addition, covert channels are addressed. The TCB must be carefully structured into protection-critical and non-protection-critical elements. The TCB interface is well defined and the TCB design and implementation enable it to be subjected to more thorough testing and more complete review. Authentication mechanisms are strengthened, trusted facility management is provided in the form of support for system administrator and operator functions, and stringent configuration management controls are imposed. The system is relatively resistant to penetration.</td>
</tr>
<tr>
<td>Class (B3): Security Domains</td>
<td>The class (B3) TCB must satisfy the reference monitor requirements that it mediate all accesses of subjects to objects, be tamperproof, and be small enough to be subjected to analysis and tests. To this end, the TCB is structured to exclude code not essential to security policy enforcement, with significant system engineering during TCB design and implementation directed toward minimizing its complexity. A security administrator is supported, audit mechanisms are expanded to signal security-relevant events, and system recovery procedures are required. The system is highly resistant to penetration.</td>
</tr>
<tr>
<td>Class (A1): Verified Design</td>
<td>Systems in class (A1) are functionally equivalent to those in class (B3) in that no additional architectural features or policy requirements are added. The distinguishing feature of systems in this class is the analysis derived from formal design specification and verification techniques and the resulting high degree of assurance that the TCB is correctly implemented. This assurance is developmental in nature, starting with a formal model of the security policy and a formal top-level specification (FTLS) of the design. In keeping with the extensive design and development analysis of the TCB required of systems in class (A1), more stringent configuration management is required and procedures are established for securely distributing the system to sites. A system security administrator is supported.</td>
</tr>
</tbody>
</table>
The NCSC built an infrastructure around the TCSEC in order to respond to its mission assigned by DOD Directive 5215.1. The evaluation criterion responded to a specific mission requirement but was not complete in and of itself. The NCSC also established a strong research and development office, a criteria and standards office, and an evaluation office to meet other missions assigned to it. The R&D effort invested time and resources in promoting advances in products and tools that could support the advance of trusted systems. Examples included strong support of multilevel secure database research, hardware based access controls, and secure operating system architectures.

The evaluation component of the NCSC was charged with evaluating commercial products against the criteria specified in the TCSEC standard and in maintaining an “evaluated products list” or EPL. A related endeavor was known as the rating and maintenance program or RAMP, which was a strong process involving certification for commercial vendors to modify their evaluated products (necessary because of normal software maintenance) and yet retain the original evaluation level. This office was populated with a very strong technical staff that worked closely with commercial technical staffs to evaluate products at the source code level as well as accompanying documentation. The evaluation process was, in concept, very straightforward and accomplished in three phases (a preliminary product evaluation, a formal product evaluation, and entry onto the evaluated products list). The preliminary product evaluation was characterized as “informal dialogue” to scope the level of effort and to make sure that all parties understood the evaluation process and its requirements. Target evaluation objectives and evaluation concerns were all addressed during this process. Once both the commercial enterprise and the NCSC evaluators decided to move forward
with the actual evaluation, they entered a formal product evaluation phase during which appropriate non-disclosure agreements and memoranda of understanding were executed. During this phase, the product was subjected to intense scrutiny and a publicly available final report was created along with a rating. The product was then entered onto the EPL at the rating level assigned. As normal software maintenance occurred over time, the product was maintained at its evaluated level by the corporate technical staff through the RAMP program as described earlier in this section.

3.5.2 Experience with the Orange Book

The evaluation process quickly became a bottleneck to the overall objective of third party evaluation and ranking. Not only was the process exceptionally time consuming, but it also required an experienced evaluation staff possessing technical skills that were in high demand in industry as well as in the federal government. Many evaluators were offered much more lucrative positions in industry while replacement personnel became increasingly difficult to find. While backlogs grew, the federal government continued to press for purchase of evaluated products. Mandates such as “C2 by 1992” became unachievable because policy was ahead of reality in terms of ability to produce products and get them evaluated in time for necessary purchases. Prices for those products that were evaluated were often not competitive with non-evaluated similar software. Waivers to government policy promoting evaluated products were often granted so that vendors that had invested heavily in the TCSEC process felt that they had not been able to recoup their investment and that the government was not following through with its earlier commitment to buy the products produced to the TCSEC specifications. Overtime, the TCSEC and the NCSC became less relevant and
other approaches began to surface. Challenges to the TCSEC approach came both nationally and internationally. In 1989, the U.S. Air Force began a program of product evaluation at their facility in San Antonio, Texas as a service to Air Force customers. This was partially a reaction to the bottleneck process at NSA as well as in response to a need for evaluation of products other than those submitted to NSA. At almost the same time, other countries began to publish evaluation criteria of their own which differed substantially in content and approach from the TCSEC. Other nations with evaluation interest and emerging criteria included Canada (Canadian Trusted Computer Product Evaluation Criteria or CTCPEC), Germany (IT –Security Criteria), the Netherlands, the United Kingdom, and France. The European efforts quickly joined together as what became know as a “harmonized criteria” in 1990, while the U.S. and Canada maintained their approach separately. The European harmonized approach became known as the Information Technology Security Evaluation Criteria (ITSEC) in 1991, and varied somewhat substantially from the North American approach. Whereas the TCSEC primarily addressed government systems and combined assurance and functionality into one rating, the ITSEC addressed commercial and government security and separated functionality from assurance. Both communities, North American and European, recognized that software manufacturers could not afford to build trusted products to different standards and began efforts to explore how they might couple the criteria in a way that the international and commercial communities could accept them. Initial efforts were directed at coming up with equivalence correspondence mappings so that a system or product rated by the European process could be viewed as some equivalent TCSEC class. These endeavors were never successful and were subject to much valid criticism.
In 1992, discussions began to merge the two approaches into a “Common Criteria” that the international community – both government and industry – could accept. Work in earnest followed a year or two later with representation from Canada, France, Germany, Netherlands, United Kingdom, and the U.S. While many draft criteria were produced for comment, the first true version of the Common Criteria was published as version 1.0 in January 1996. Taking into account comments, reviews, and experience with this initial version, a revised version 2.0 was released in May 1998. A modification of this second version, version 2.1, was adopted by the International Standards Organization (ISO) as an international standard in 1999 (ISO Standard 15408).

3.5.3 The Common Criteria (CC)

Today, ISO Standard 15408 is the recognized computer security evaluation and rating criteria internationally and the TCSEC was formally retired in 2001 by the U.S. government. Following the development of the Common Criteria, the National Institute of Standards and Technology (NIST) and the National Security Agency, in cooperation with the U.S. State Department, worked with the CC Project to produce a mutual recognition arrangement for IT security evaluations. In October 1998, after two years of negotiations, government organizations from the United States, Canada, France, Germany, and the United Kingdom signed a historic recognition arrangement for Common Criteria-based IT security evaluations. The “Arrangement” (officially known as the Arrangement on the Mutual Recognition of Common Criteria Certificates in the field of IT Security), was a significant step forward for both government and industry in IT product security evaluations. The U.S. government and its partners in the Arrangement agreed to the following objectives with regard to common evaluations:
- Ensure that evaluations are performed to high and consistent standards and are seen to contribute significantly to confidence in the security of those products and profiles
- Increase the availability of evaluated products for national use
- Eliminate duplicate evaluations between the signatories
- Improve the efficiency and cost-effectiveness of security evaluations and the certification/validation process

In October 1999, Australia and New Zealand joined the Mutual Recognition Arrangement increasing the total number of participating nations to seven. Following a brief revision of the original Arrangement to allow for the participation of both certificate-consuming and certificate-producing nations, an expanded Recognition Arrangement was signed in May 2000, at the 1st International Common Criteria Conference by Government organizations from thirteen nations. These include: the United States, Canada, France, Germany, the United Kingdom, Australia, New Zealand, Italy, Spain, the Netherlands, Norway, Finland, and Greece. The State of Israel became the fourteenth nation to sign the Recognition Arrangement in November 2000. The Common Criteria continues to this day to gain in acceptance and several other nations are actively considering its adoption (e.g., Russia, China, and Japan).

The Common Criteria represents a departure from the TCSEC approach and is more closely related to the approach developed by the European community. A very brief overview will be presented here, but for a more detailed review of this document and its processes the interested reader is directed to http://csrc.nist.gov/cc/, http://www.commoncriteriap.org/, and http://niap.nist.gov/. The CC is a lengthy document divided into three parts. Part 1 provides background information on the criteria itself, an
overview of its processes, and serves as a general reference. Part 2 addresses functional requirements and assists users in formulating statements of requirements for secure systems and assists developers in interpreting those requirements. Similarly, Part 3 addresses assurance requirements. Note that assurance and functionality have been separated in this document and are no longer coupled as they were in the TCSEC. The CC also introduced several new terms that where important to its processes. First is that of the Protection Profile (PP). A protection profile is an implementation independent requirements document that specifies a need. A consumer can use requirement statements from Parts 2 and 3 to describe the functionality and assurance needed in a product. Additionally, the PP contains a statement of the security problem to be solved by the IT product (which may include specific assumptions that may be made concerning the operating environment, the anticipated threats, and organizational security policies). The PP, although a requirements document, can be submitted for evaluation under the CC and receive a rating. Once rated, it can be placed on the list of evaluated products so that others can make use of it. There was no equivalent to this in the TCSEC process. An IT product that is to be evaluated is known as a Target of Evaluation (TOE). Actually, a TOE can be a complete product or a part of a product or system. It includes the product itself and all associated documentation. In order to have a TOE evaluated under the CC process, a Security Target (ST) must be created. For the most part, the ST follows the same specified format as the PP with the exception that the ST references a specific PP and contains a description (claims) of how the TOE meets the requirements of a PP or where it falls short of doing so. Whereas the PP is a generic requirement – the ST is specific to a particular TOE and makes specific claims as to its assurance, its
functionality, and its compliance with the PP requirements. An evaluation, therefore, under the terms of the CC would require the ST, the set of evidence about the TOE and the TOE itself. The result of the evaluation would be confirmation that the ST is satisfied by the TOE. More simply stated, the PP is the end user requirement, the TOE is the product, the ST is the claim that the product meets the need, and the evaluation is the 3d party review that the claim is correct.

Under the terms of the CC process, a certified private laboratory accomplishes the evaluation of the actual product. Each signatory to the CC has a government oversight body that validates laboratories as being compliant with CC standards. Evaluations conducted at any laboratory certified by a signatory to the CC are acceptable (at the first four levels of evaluation only) to all nations that participate in the CC recognition agreement. Evaluations are paid for by the product manufacturer as agreed to by the vendor and the laboratory. This is a major departure from the early U.S. TCSEC procedure that offered a single evaluation facility that was largely funded by the government.

The evaluation scheme consists of seven levels – evaluation assurance level (EAL) 1 through 7. Each level is only an assurance level and does not imply any specific mechanisms since the CC decoupled mechanisms and assurance. Their meaning is summarized below as taken from the Common Criteria introductory brochure

- EAL 1: Functionally tested. Used where some confidence in correct operation is required, but the threats to security are not viewed as serious. The evaluation at this level provides evidence that the TOE functions in a manner consistent with its documentation, and that it provides useful protection against identified threats.
- EAL 2: *Structurally tested*. Evaluation at this level involves a review of design information and test results. This level may be appropriate where developers or users require a low to moderate level of independently assured security in the absence of ready availability of the complete development record (e.g., when securing legacy systems).

- EAL 3: *Methodically tested and checked*. This level is applicable where the requirement is for a moderate level of independently assured security, with a thorough investigation of the TOE and its development. An EAL 3 evaluation provides an analysis supported by testing based on “gray box” testing, selective independent confirmation of the developer test results, and evidence of a developer search for vulnerabilities.

- EAL 4: *Methodically designed, tested and reviewed*. This is the highest level at which it is likely to be economically feasible to retrofit an existing product line with security. It is applicable to those circumstances where developers or users require a moderate to high level of independently assured security and there is willingness to incur some additional security specific engineering costs.

- EAL 5: *Semiformally designed and tested*. This is applicable where the requirement is for a high level of independently assured security in a planned development, with a rigorous development approach, but without incurring unreasonable costs for specialized security engineering techniques. Assurance is supplemented by a formal model and a semiformal presentation of the functional specification and high level design, and a semiformal demonstration of correspondence. A search for
vulnerabilities that includes resistance to penetration attacks and a covert channel analysis is also required.

- **EAL 6: Semiformally verified design and tested.** This EAL is applicable to the development of specialized TOEs for application in high risk situations where the value of the protected assets justifies the additional costs. The evaluation provides an analysis, which is supported by a modular and layered approach to design. The search for vulnerabilities must ensure resistance to penetration by attackers with high attack potential. The search for covert channels must be systematic. Development environment and configuration management controls are enhanced at this level.

- **EAL 7: Formally verified design and tested.** Applicable to the development of security TOEs for application in extremely high risk situations, and/or where the high value of the assets justifies the higher costs. For an evaluation at this level, the formal model is supplemented by a formal presentation of the functional specification and high level design showing correspondence. Evidence of developer “white box” testing and complete independent confirmation of developer test results are required. As a practical matter, a TOE at EAL 7 must minimize design complexity and have tightly focused security functionality amenable to extensive formal analysis.

There is a distinct break between EAL 1 through 4 versus EAL 5 through 7. EAL 4 and below can generally be achieved by retrofitting existing products and systems and can be done so economically. Evaluation certification of an IT product by any laboratory certified by a signatory of the CC at EAL1 through 4 is acceptable in any other signatory nation. Products certified at EAL 5 through 7 must be certified by the nation that uses them in its government systems. At EAL 5 and above, specialized security engineering
techniques are required and retrofit of existing products is generally not possible. A complete list of certified products can be found at http://www.commoncriteria.org/. The reader will note that most evaluated products lie in the EAL 1 through 4 range (see table 3).

One of the objectives of the CC effort was to maintain a backwards compatibility with the TCSEC (recall that it had been used by the U.S. for over ten years prior to the arrival of the CC) and the European ITSEC. This was made necessary due to the heavy investment that the software industry had made in producing evaluated products and in order for government agencies to remain compliant with specific regulatory guidance. The general equivalency between CC, TCSEC, and ITSEC evaluations is given in Table 2 as published by the CC working group. Although commercial vendors of trusted products sometimes cite this table, one needs to be aware that many valid arguments can be made that the mappings are not exact and that no direct comparisons can be made. It serves as a general guideline only.

<table>
<thead>
<tr>
<th>Common Criteria</th>
<th>TCSEC</th>
<th>ITSEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>D: Minimal Protection</td>
<td>E0</td>
</tr>
<tr>
<td>EAL1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EAL 2</td>
<td>C1: Discretionary Access Protection</td>
<td>E1</td>
</tr>
<tr>
<td>EAL 3</td>
<td>C2: Controlled Access Protection</td>
<td>E2</td>
</tr>
<tr>
<td>EAL 4</td>
<td>B1: Labeled Security Protection</td>
<td>E3</td>
</tr>
<tr>
<td>EAL 5</td>
<td>B2: Structured Protection</td>
<td>E4</td>
</tr>
<tr>
<td>EAL 6</td>
<td>B3: Security Domains</td>
<td>E5</td>
</tr>
<tr>
<td>EAL 7</td>
<td>A1: Verified Design</td>
<td>E6</td>
</tr>
</tbody>
</table>

It remains to be seen how effective the CC scheme will be over time. At the present, it seems to be gaining acceptance and strength. As of this writing, there are
approximately fifty IT products on the evaluated list, which have been certified between 1997 and 2002 and several others undergoing evaluation. Table 3 depicts the number of products by year and evaluation level achieved. Clearly, early evaluations were concentrated on lower level assurance (EAL 1 and 2) while in more recent years higher levels of assurance have begun to appear on the list (EAL 3 to 5). While some of this trend is likely due to an initial effort by vendors to obtain the quickest and lowest cost evaluation, it also reflects a growing willingness to invest in the process and the desire to produce products that are more trustworthy than in the past. It is also interesting to note that evaluations at the very highest levels of EAL 6 and 7 are missing which may be a reflection on a lack of necessary software engineering tools to perform adequate formal analysis, verification, and proofs of correctness for large software based systems or perhaps it is an indication that vendors are not yet willing to risk the high cost of development of such systems until they are more confident of a return on investment. In order to achieve a multilevel security capability, IT products and systems will need EAL 6 and 7 levels of assurance. To promote the use of CC evaluated products within the U.S. government, the National Security Telecommunications and Information Systems Security Policy (NSTISSP) number 11 was issued in January 2000 which requires that preference be given to acquisition of commercial off-the-shelf (COTS) information assurance (IA) products and IA-enabled IT products which have been evaluated in accordance with the Common Criteria, the National Information Assurance Partnership (NIAP), or NIST Federal Information Processing Standard (FIPS) programs. The policy further requires that as of July 1, 2002, acquisition of such products will be limited to that that have been evaluated. It remains to be seen whether or not this policy will indeed
have the desired effect to promote an increase in the number of evaluated products and their use – or whether the policy’s waiver process will be used extensively to circumvent it. In today’s environment, with a necessity to establish a strong defense in depth strategy, there simply is not enough diversity of product on the evaluated list from which one could construct a total defense so the waiver process will have to be used for valid reasons. Additionally, for the CC process to be a long-term success, it will need to gain acceptance in the industrial (non government) communities – an objective that has not been met today.

<table>
<thead>
<tr>
<th></th>
<th>EAL 1</th>
<th>EAL 2</th>
<th>EAL 3</th>
<th>EAL 4</th>
<th>EAL 5</th>
<th>EAL 6</th>
<th>EAL 7</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1998</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>1999</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>2000</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>2001</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>2002</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>TOTAL</td>
<td>7</td>
<td>12</td>
<td>14</td>
<td>15</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>49</td>
</tr>
</tbody>
</table>

3.6 Other Notable Legislation

While some legislation and government guidance has been presented already, several key laws and directives were omitted. A reasonably comprehensive treatment of computer security mandates and legislation can be found in the Russell and Gangemi text titled Computer Security Basics, Appendix B.

The importance of the Brooks Act of 1965 has already been discussed. Although several additional mandates, directives, and laws pertaining to classified information processing and protection of signals were issued in years following the Brooks Act, it was not until 1984 that the next major computer security mandate was issued. That came in the form of National Security Decision Directive (NSDD) 145, which among other
things, created a high level government computer security oversight committee which over time (and with the help of National Security Directive 42 in 1990) created a policy body known as the National Telecommunications and Information Systems Security Committee (NTISSC). This directive also required protection for sensitive, but unclassified information for the first time and assigned to the National Security Agency the tasks of encouraging, advising, and assisting the private sector in information protection. This appeared to be a major shift in national leadership responsibility for computer security from NIST to NSA – but not without controversy. The National Security Agency had long been regarded as a closed intelligence organization with almost no public presence. The assignment of a public mission to such an organization that involved assisting the public sector in the protection of information by an organization whose intelligence role included penetrating foreign government systems seemed, to some, a bit of a conflict. This role included providing encryption algorithms to the public sector to protect data in transit – while NSA retained the role of being responsible for breaking encryption algorithms for intelligence purposes. Notwithstanding this appearance of a conflicting role – there was also a certain amount of mistrust prevalent in the public sector toward such an organization and its culture.

Responsibility for computer security at the federal level seemed to shift once more with the Public Law 100-235 (also known as the Computer Security Act of 1987), which became effective in the fall of 1988. This law assigned to NIST responsibility for computer security vulnerability assessments, standards, technical assistance, guidelines, and training and gave NSA a supporting role to NIST. The law did not apply to classified material and left responsibility for that with NSA. Additionally, the law required all
federal agencies to create a specific computer security plan for their organization and to provide computer security training for their people. While these actions might have seemed fairly basic, they were needed in that few organizations had such plans and many that did had not exercised them or updated them for the actual environment they were working in.

In 1996 Congress passed what has become known as the Clinger-Cohen Act (also called the IT Management Reform Act) that assigned to the Office of Management and Budget the responsibility for acquisition and management of IT. While much of what the Act mandated was strictly procurement related, it did have interesting side effects related to security. These included giving authority to acquire IT resources to the head of each executive agency of the government and encouraging the procurement of commercial off-the-shelf (COTS) products as preferred over initiating special developments. Most importantly, the Act required the appointment of a Chief Information Officer (CIO) in federal agencies – an office that naturally assumes the responsibility for computer security in most cases. Later in 1996, President Clinton issued an Executive Order creating a CIO council that has today assumed computer security policy and mandates to be a part of their charter. The core responsibility for carrying out the requirements of the Clinger-Cohen Act and the follow on Executive Order 13011 (establishment of the CIO Council), lies with the Office of Management and Budget or OMB which has statutory responsibility for setting policy for the security of Federal automated information systems. It implements these responsibilities through OMB Circular A-130 Appendix III, “Security of Federal Automated Information Resources” (see http://www.whitehouse.gov/omb/circulars).
Nearly simultaneous with the actions in the preceding paragraph, President Clinton issued Executive Order 13010 in 1996, creating the President’s Commission on Critical Infrastructure Protection (PCCIP). The order stated, “Certain national infrastructures are so vital that their incapacity or destruction would have a debilitating impact on the defense or economic security of the United States. These critical infrastructures include telecommunications, electrical power systems, gas and oil storage and transportation, banking and finance, transportation, water supply systems, emergency services (including medical, police, fire, and rescue), and continuity of government. Threats to these critical infrastructures fall into two categories: physical threats to tangible property ("physical threats"), and threats of electronic, radio frequency, or computer-based attacks on the information or communications components that control critical infrastructures ("cyber threats"). Because many of these critical infrastructures are owned and operated by the private sector, it is essential that the government and private sector work together to develop a strategy for protecting them and assuring their continued operation.” The significance of this EO is that it was the first to recognize the vulnerability of the Nation’s infrastructure systems and the dependence our national security places on their operation. In the words of the resulting report, “The cyber dimension promotes accelerating reliance on our infrastructures and offers access to them from all over the world, blurring traditional boundaries and jurisdictions. National defense is not just about government anymore, and economic security is not just about business. The critical infrastructures are central to our national defense and our economic power, and we must lay the foundations for their future security on a new form of cooperation between the private sector and the federal government. The federal
government has an important role to play in defense against cyber threats -- collecting information about tools that can do harm, conducting research into defensive technologies, and sharing defensive techniques and best practices. Government also must lead and energize its own protection efforts, and engage the private sector by offering expertise to facilitate protection of privately owned infrastructures.” The report listed eight infrastructures “so vital that their incapacity or destruction would have a debilitating impact on our defense and economic security.” These infrastructures and their importance were reported as:

- **Transportation** – moves goods and people within and beyond our borders, and makes it possible for the U.S. to play a leading role in the global economy.

- **Oil and gas production and storage**: fuels transportation services, manufacturing operations, and home utilities.

- **Water supply**: assures a steady flow of water for agriculture, industry, business, firefighting, and homes.

- **Emergency services**: responds to our urgent police, fire, and medical needs.

- **Government Services**: consists of federal, state, and local agencies that provide essential services to the public.

- **Banking and finance**: manages trillions of dollars – from individual accounts to support of global enterprises.

- **Electrical power**: generation, transmission, and distribution systems that are essential to all other infrastructures and every aspect of the nation’s economy.
- **Telecommunications**: includes the public telecommunications network, the Internet, computers used in homes, commerce, academia, and government, and all forms of communication that connect systems together.

The final report can be accessed at http://www.ciao.gov/resource/pccip/report_index.htm for those that are interested in further exploring the findings.

Based on the findings of the PCCIP, President Clinton signed Presidential Decision Directive (PDD) 63 on May 22, 1998. This event officially expanded the nation’s policy interest to the cyber security world and, in effect, recognized a need to couple more tightly the nation’s critical infrastructure industrial base with corresponding federal government offices and to establish a specific law enforcement focus on cyber protection and incident response. To this end, PDD-63 formally assigned lead government agencies to specific industrial sectors and for what was referred to as “special functions”. Table 4 depicts the agency/sector partnering.

### Table 4

**PDD-63 Lead Agency for Sector Liaison**

<table>
<thead>
<tr>
<th>LEAD GOVERNMENT AGENCY</th>
<th>CRITICAL INFRASTRUCTURE SECTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commerce</td>
<td>Information and communications</td>
</tr>
<tr>
<td>Treasury</td>
<td>Banking and Finance</td>
</tr>
<tr>
<td>Environmental Protection Agency</td>
<td>Water Supply</td>
</tr>
<tr>
<td>Transportation</td>
<td>Aviation, Highways, Mass transit, Pipelines, Rail, Waterborne commerce</td>
</tr>
<tr>
<td>Justice/FBI</td>
<td>Emergency law enforcement services</td>
</tr>
<tr>
<td>Federal Emergency Management Agency</td>
<td>Emergency fire service, Continuity of government services</td>
</tr>
<tr>
<td>Health and Human Services</td>
<td>Public health services, including prevention, surveillance, laboratory services and personal health services</td>
</tr>
<tr>
<td>Energy</td>
<td>Electric power, Oil and gas production storage</td>
</tr>
</tbody>
</table>
As of the writing of this Chapter, the lead agencies are as established by PDD-63, but the terrorist attacks of September 11, 2001 and the subsequent establishment of the Office of Homeland Security will likely change this assignment as responsibilities are realigned in the federal government. The President’s decision required each lead agency to designate one individual of Assistant Secretary rank or higher to be the sector liaison official and to coordinate/cooperate with the private sector in addressing problems related to critical infrastructure protection. In addition to the sector liaisons, the PDD also established lead agencies for “special functions”. Special functions were defined as those related to critical infrastructure protection that *must* be chiefly performed by the federal government (e.g., national defense, foreign affairs, intelligence, and law enforcement). The most important aspect of the special function lead is that they have responsibility for coordinating all the activities of the federal government in their assigned area. The lead agencies and their designated special functions are:

- Department of Justice/FBI: Law enforcement and internal security
- Central Intelligence Agency: Foreign intelligence
- Department of State: Foreign affairs
- Department of Defense: National defense

The above is a high level overview of the structure established by the federal government. Other functions, such as the role of a National Coordinator for Security, Infrastructure Protection and Counter-Terrorism; the creation of a National Infrastructure Protection Center (NIPC); and the research and development role for the Office of Science and Technology Policy are all discussed in the PDD. The full version of PDD-63 can be found at http://www.terrorism.com/homeland/PDD-63.pdf for the interested
reader. While the objectives of the PDD were quite lofty, actual progress has been slow and real results are few. The greatest contribution, however, was its focus on the problem as a national policy issue and the formal structure it created.

As a reaction to the terrorist attacks of September 11, 2001 a new law know as the Patriots Act was hurriedly pushed though Congress and signed by the President on October 26, 2001. The bill is 342 pages long and makes changes, some large and some small, to over 15 different statutes. The government may now monitor web surfing activities of Americans, including terms entered into search engines, by merely convincing a judge that the monitoring could lead to information that is "relevant" to an ongoing criminal investigation. The person monitored does not have to be the target of the investigation. The judge must grant this application and the government is not obligated to report to the court or tell the monitored subject what has been done. The law also makes two changes to increase how much information the government may obtain about users from their Internet service providers (ISPs) or others who handle or store their online communications. It allows ISPs to voluntarily hand over all "non-content" information to law enforcement with no need for any court order or subpoena. Second, it expands the records that the government may seek with a simple subpoena (no court review required) to include records of session times and durations, temporarily assigned network (I.P.) addresses, means and source of payments, including credit card or bank account numbers. Lastly, and most important, the Patriot Act makes a tie between terrorism and computer crime. This piece of legislation has the potential to become a defining point in combating computer crime, but will certainly be challenged in the courts over the next few years.
3.7 Worms, Viruses and other Malevolent Code

No historical treatment of computer security would be complete without some discussion of the impact that malicious code has had on the computing community. Malicious code is software that is created to intentionally violate policy or desired rules of behavior on your system – often with destructive intent. Code run on any computer system executes at some permission level – normally either the permission level of a particular user or at “system level”. Access controls normally limit what actions user level permissions can take, while system level permission is generally unrestricted. The challenge of the author of malicious code is to have the code run at the highest permission level possible so that it activities are not restricted. Nonetheless, any malicious code run at a specific user’s permission level can certainly damage, destroy, or compromise any information assets owned by that user and that alone is generally considered a significant threat.

Malicious code exists in many varieties and is classified in several ways. The intent of this section is not to define a taxonomy of malicious code or to present precise definitions, but rather to provide the reader with a sense of the types of malicious code, the basic way they operate and to disclose some noteworthy incidents. We include the term mobile code in this discussion to refer to code that is downloaded from a server to a client machine and executes on the client. Mobile code is not necessarily bad and in the majority of cases it performs a useful and desired function, but the potential exists for such code to be malicious in the client’s environment.

Malicious code either operates on its own (independent) or requires a host program (dependent). Some such code has the ability to reproduce (propagate) while
others may not. The generally accepted types of malicious code are trap doors, Trojan horses, logic or time bombs, viruses, and worms. There are many more names, nuances, and descriptions of such code in the literature, but these are sufficient for this discussion. We begin with a brief description of each type of malicious code followed by a discussion of some historical landmark incidents and end with some thoughts on how one defends against such attacks today.

Malicious code is written intentionally to cause harm or at least mischief. It is different from legitimate software errors that may result in harm or system failure. While the end result is the same, the cause and defenses against it are different. Intent is the differentiator.

- **Trap door** – a secret, undocumented entry point into program software that can be used to grant access without going thought the normal access authentication process. This might be, for example, software that recognizes a certain sequence of keystrokes or accepts a certain key word as input to turn control over to the attacker. This code does not reproduce itself nor can it “infect” other code. It is a specific entry point into executable code that is generally undocumented. Programmers often insert trap doors in code under development to assist in testing or as a shortcut in debugging. This practice is discouraged in most software development activities, yet it does occur. Occasionally, the programmer forgets to remove the trap door and it is discovered by others who exploit it for malicious purposes.

- **Trojan horse** – a computer program with an apparent or actual useful function that contains additional hidden functionality that surreptitiously exploits the legitimate authorizations of the invoking process to the detriment of security. These are often
found in a useful utility program or game that when executed may also delete files or change file permissions or corrupt data. An example of this kind of attack might be a free computer game that when executed not only provides a game function for the user, but also accesses the user’s files and writes them off to another location that the attacker can access later. A Trojan horse does not replicate or infect other programs.

- **Logic or time bombs** – a computer program that contains a malicious function and lies dormant in a computer system until triggered by a set of logical events (logic bomb) or a specific time/date combination (time bomb). Such bombs have been a nuisance for a very long time in computing. An example of a logic bomb might be code written into a financial system that destroys all financial records maintained by the system if a certain social security number is not present in the payroll file. This might be a tactic employed by a disgruntled programmer to guard against being fired. An example of a logic bomb might be code that deletes files on Friday the thirteenth, April Fool’s Day, or Halloween for example. Logic or time bombs by themselves do not replicate or infect other files.

- **Viruses** – malicious software that is embedded within other code (normally another computer application program) or is hidden in a boot sector of a disk. It normally consists of three parts – a mission, a trigger, and a propagation (replication) component. A virus does not exist on its own and requires a host in order to execute and reproduce. When a legitimate (but infected) program executes, it will also execute the virus code unknowingly. The virus code will generally copy itself (infec) to another software program on the computer or imbed itself in memory and copy itself to every program that runs on that machine. After replicating itself, it may
execute a specific function (mission) such as delete files, copy files, or modify data. In some cases, the virus has a trigger (logical or time condition) that it checks first before executing the mission. Similar to the logic or time bomb approach, this might be a trigger that looks for a specific program being ran before it executes or it may look for a specific time of day, month or year (e.g., Friday the thirteenth). Viruses are sometimes referred to as “self-replicating Trojan horses”.

- **Worms** – malicious software that does not need a host to survive. Worm software is written to be independent code that replicates and travels thought networks infecting machines. Like a virus, it can have a mission, trigger, and propagation component. It is designed, however, to attack networks and distributed systems.

- **Mobile Code** – came into existence as an efficiency technique associated with client server architectures and Internet usage. This is code that exists on a distant server and when a client connects, it is downloaded and executes on the client machine. There are many specific techniques used to accomplish this – some more risky to the user than others. Examples might include Java Applets, Active X, plug-ins, and JavaScript. While knowledgeable users have the ability to restrict or prohibit such code from running on their machine, in practice few actually do. The vast majority of mobile code implementations are beneficial and not malevolent but the technique itself does represent a vulnerability that can be exploited.

The use of malicious code has a rich and interesting history. It has been reported that even John von Neumann, often considered the father of the modern day computer, once published an article in 1949, titled "Theory and Organization of Complicated
Automata\textsuperscript{17} - a report which dealt with what he referred to as self reproducing automata – an early description of what we now call a virus. Early scientists and computer technical staffs often used these techniques in games designed to prove their skill. A game originally called \textit{Darwin} and later \textit{Core Wars} was a contest between `assembler' programs in a machine or machine simulator, where the objective was to kill your opponent's program by overwriting it. The game was devised and played by Victor Vyssotsky, Robert Morris Sr., and Doug McIlroy in the early 1960’s while working at Bell Labs. Most such exploits were harmless pastimes for technical wizards of the day. During the late 1960’s and into the mid-1970’s government penetration testing used the technique of a Trojan horse to exploit operating systems and achieve access. In the early 1980’s, researchers at Xerox tried to create what we would refer to as a “worm” today that had no malicious intent and was designed as a maintanence program. Due to a flaw in the program logic, this Xerox worm\textsuperscript{18} accidently caused a denial of service issue. Essentially, there was very little of any malicious activity although the potential was there to exploit.

This changed dramatically in the early 1980’s when a researcher by the name of Dr. Fred Cohen, then a professor of Computer Science and Electrical Engineering at Lehigh University, began a research effort focusing on computer viruses. The results of his work were published widely, but the initial publication came in a paper dated August 31, 1984 titled simply “Computer Viruses” and delivered at the 7\textsuperscript{th} Department of Defense/National Bureau of Standards Computer Security Conference\textsuperscript{19}. Within this paper, Dr. Cohen described the technique of creating a virus, general experiments with their propagation, concluded that virus detection by analysis is undecidable, and that
countermeasures are extremely difficult to employ. Interest and experimentation in this technique quickly became widespread and continues into the present. Incidents have been reported involving computer viruses used for destructive purposes, industrial sabotage, as a terrorist tool, as an advertising ploy, and simply for the purpose of creating one. Over time as technology changes, the techniques of using a virus have evolved and have been adapted to macro languages (e.g., in Microsoft products like Excel or Word), some viruses have been created that change their appearance or signatures (called polymorphic viruses), and others have introduced code to help avoid detection and removal. While viruses have continued to plague the computing community they have largely become more of a nuisance than the disaster they once were considered to be.

A major incident occurred in November of 1988, that is arguably the single most published computer security event and the one incident that more than any other, focused international attention on computer security and the fragility of operating systems and networks. While the incident is described in great detail in many publications since, the interested reader may wish to obtain a copy of the June 1989 issue of Communications of the ACM (volume 32, number 6) for a good synopsis of the attack. On November 2, 1988 a graduate student at Cornell University, Robert Tappan Morris, released a computer worm that he had created, into the Internet. Within hours the worm had penetrated thousands of computers and had caused many to fail due to a flaw in the worm that resulted in it consuming too many system resources and causing a denial of service (DOS) problem. The worm itself exploited several flaws in Unix operating systems that were, for the most part, well known in the technical community. It involved password cracking, exploitation of design flaws, and exploitation of design “features” to replicate
and travel the Internet. Mr. Morris was identified within days of the incident and was eventually found guilty of violating section 1030 of U.S. Code Title 18 (the 1986 Computer Fraud and Abuse Act). He received a $10,000 fine, a suspended jail sentence with a community service obligation of 400 hours. This single incident not only focused community attention on computer security, but it also started a series of events to change the law, revisit the study of computing ethics, and the creation of Computer Emergency Response Teams (CERTs). While many of the details of the attack are omitted here, it is important to realize that the same kind of attack can and does occur today. Many of the vulnerabilities exploited by the 1988 worm still exist.

Defense against malicious code and mobile code is today an important consideration. A total defense is not possible but mitigation strategies are possible and recommended – primarily against viruses, worms, and malevolent mobile code. Many good virus detection software products are available on the market today and it is important that users have one and keep it updated. Most virus scanners look at incoming email, web transactions, and software distribution media (disk or CD), and resident software for the presence of viruses or worms. Since these products work on the basis of recognizing known viruses and variants of them, it becomes important to update the virus scanner code frequently (one a week or so). New or re-designed viruses are discovered frequently and their recognition patterns need to be added to a user’s virus detection system. This is normally accomplished over an Internet connection to the vendor’s web site. Some worms can be discovered by virus scanning software, but not all. Having a firewall in place, which isolates a network, is helpful as a worm defense mechanism. Keeping operating systems up-to-date with the latest patch version is also an important
defense. During the early 2000’s a new software product began to appear known as operating system behavior based tools. These products are useful as a defense against virus or worm attacks that are new or have not yet been added to a virus scanner. The idea behind this approach is to establish user-defined rules of un-allowed behavior at the operating system kernel level so that if executing code attempts to perform some malicious function in violation of a rule, it is prevented. Finally, policy, procedure, and training employed by the organization helps to avoid activity that can introduce a virus and assists employees in identifying when a virus or a worm has been introduced. Web browsers today have security settings that help to act as a deterrent to malicious mobile code. By accessing browser security settings, a user can allow or disallow various forms of mobile code. Additionally, many reputable software producers “sign” their mobile code with a certificate that is displayed on the client machine so that the user can accept or reject the code based on knowing where it came from. Certified code is generally reasonably safe, while code from unknown sources may not be. Many organizations today with centrally managed networks, are establishing policy toward mobile code and implementing that policy through system settings established and maintained by the systems administrator.

3.8 Summary and Concluding Comment.

This section has presented a historical treatment of computer security with concentration on many events that the author considered important over the past thirty years. This is by no means a complete historical and topical treatment, but it does show progress over time and an increasing understanding of the problem. The motivations for computer security have been presented, design principles introduced, attempts to mandate
security through law and policy were discussed, and significant events were related. In each case, the historical event was related to today’s approach (as in movement from the orange book to the common criteria). It was the intent of this section to show advances over thirty years of work in the computer security/information assurance area, yet also demonstrate that much of the basic problem that one must grapple with in security has not really changed although technology itself has changed substantially. As Dr. Ware and the Defense Science Board so effectively pointed out in their landmark 1970 report, securing an information system is a difficult process that involves more than technical solution sets. It remains a combination of effective policies, procedures, strong (high assurance) software products, and defensive in depth architectures all coupled with trained and trusted employees.

4. Today’s Treat and Countermeasures

4.1 Overview and Awareness.

While the actual problem of securing computing systems and networks remains largely unsolved, substantial progress has been made over the past thirty years. Even the realization that perfect security is unobtainable must be considered a form of progress in understanding and allows the community to factor this risk into policy and procedure in today’s information infrastructure. Improvements can be seen in the number of security products available today, the strength of those products, the number of security engineers, training of employees, strong internal policies, stronger operating systems, better networking procedures, and more effective laws. Awareness alone is not sufficient to fully address this problem but does help employees understand and adhere to policies and helps managers support the IT budgets necessary to safeguard systems. In present day
security a security engineer has a broad experience base on which to examine the
effectiveness of products and has the advantage of an evaluation process that many
products have been exposed to through either the old Trusted Computing System
Evaluation Criteria program (aka, the Orange Book process) or the newer, more current
ISO Standard 15408, or Common Criteria. In short, today’s security engineer knows
more about the problem, the threat, and what comprises sufficient defenses than was
generally known twenty or thirty years ago. On the negative side, the problem of
knowing how to compose together a set of products that offer sufficient protection for a
system is still a matter of individual engineering expertise and, to a certain extent, an art
more than a science. Measurement of a protection structure in terms of “how much”
security is achieved by a specific architecture is not possible today. It is also often not
possible to shape a realistic return on investment strategy when one is faced with
convincing the financial watchdogs that a particular security product or capability should
be purchased for an organization.

The ability to secure systems would appear to have not advanced much – but such
an observation would be deceptive at best. First, the field of computing itself is roughly
only 50 years old. The interest in securing such systems has only been the focus of any
serious attention for the last 30 years. Security of desktop machines at home and office
has only been an issue for less than 20 years. While these are relatively short periods of
time in the field of scientific advances, one must also consider that the field of computing
has been advancing rapidly while the investigations into its security aspects continued. A
more succinct way of stating this might be to say that the provision of computing security
has been a moving target with technology moving ahead faster than the security solution
set seems to move. On top of advancing technology has been the issue of unreliable software. Major vulnerabilities exist in software today that allow outsiders to acquire unauthorized access or elevate their privileges, cause denial of service attacks, or affect the integrity of stored and processed data. Rather than become more secure over time, software seems to have become less secure. While software engineering principles are reasonably well understood, they are simply not practiced well today in many organizations nor has the community that relies on reliable software been very insistent on reliable code. For the reader interested in pursuing this observation a bit further, an article\textsuperscript{20} addressing this topic written by Charles Mann in Technology Review is worth reviewing. An interesting observation made by Mr. Mann is that “Microsoft released Windows XP on Oct 25, 2001. That same day, in what may be a record, the company posted 18 megabytes of patches on its Web site: bug fixes, compatibility updates, and enhancements. Two patches fixed important security holes. Or rather, one of them did; the other patch didn’t work.” This, as well as many other such episodes, attests to the fact that several dynamics are at play here – they include the expanding complexity of software systems today (Windows XP has approximately 45-million lines of code) versus systems of several years ago; the lack of programmer quality focus (a focus which seems to be decreasing over the years); and the lack of appropriate software engineering tools to prove code correct (verification and validation tools). The overall quality of software is an important consideration in the overall security of a system. The evaluated products process sponsored by the National Institute of Standards and Technology (NIST) implementing the ISO Standard 15408 helps to improve confidence in the quality as well as the trustworthiness of software.
4.2 Products and Procedures

![Diagram of Typical Architecture for an Information Protection Network]

Figure 1: Typical Architecture for an Information Protection Network

Early researchers believed that access control by the operating system was the key technical focus to providing trustworthy computing and that if access control could be guaranteed to conform to some established security policy, then enforcement by the operating system could insure that no unauthorized access of a user to data could occur. While this model may have been satisfactory in a standalone, mainframe world – it could not scale up to technology changes, which included networking, database systems, wireless technology, high performance computing clusters, electronic commerce, and other advances. While this statement is not meant to diminish the role of strong access control in any way, it is meant to show that there are issues beyond access control that need to be solved as well in order to truly establish a secure environment. Today, a defense in depth strategy is necessary in order to provide security in a networked
environment. An example of such a strategy at a high level is shown at Figure 1 where a typical architecture for an information protection network (IPN) is shown. When operating in a networking environment, a security engineer may be charged with insuring that the assets of the protected company network are secure from attacks from the outside world over Internet connectivity. To do so, the engineer will weave together an intermediate network of protection mechanisms and products that is fitted between the protected network and the open Internet. This is sometimes referred to as a Demilitarized Zone or DMZ. The IPN can process and analyze incoming and outgoing traffic in this “buffer zone” and add a significant degree of protection for the internal user. Using this IPN template in Figure 1, a security engineer can make adjustments by adding or deleting products that lead to the engineered solution needed. For example, the engineer may decide that a web proxy sever is actually not needed and delete that device based on a risk analysis or vulnerability assessment performed on the system being protected. Similarly, the engineer may decide that a network filter product is needed to filter outgoing and incoming emails looking for unauthorized communications. A wide variety of products can be accommodated in the IPN, but the final solution set is determined on a situation specific basis by a trained security engineer who uses personal knowledge and skill to design it and tweak it over time. Note that the IPN does include a strong access control component at the network level – an important fundamental consideration. The IPN also, however, accommodates other security concerns such as privacy of network addresses, proper policy enforcement, malicious code detection, remote authentication, and other important functions. The security engineer will also be concerned with products that operate at the host level (which may be a desktop machine, a large central
server, a mainframe, a traveling laptop, or some combination of these). Certainly the trustworthiness of the OS is important, but so are many other features that may result in product selection for each machine. Figure 2 depicts a template of concerns for a specific host that once again may be tweaked and modified by the engineer to meet a particular risk or address a specific vulnerability. Access control and identification and authentication (I&A) are generally included in the operating system, but stronger I&A may be required. In such a case, the engineer may need to choose from an array of products that might be useful to strengthen this function. That may include choosing a smart card product or perhaps a biometric device (e.g., fingerprint scanner, voice identification, or retina scanner). Similarly, products to combat intrusions, malicious code, or misuse of the computing resource may need to be added (e.g., a personal firewall, a virus scanner, and an network filter). If a wireless card is present and used, special encryption software may need to be installed so that a virtual private network (VPN) can be accommodated.
In general, the engineer must insure that protection coverage is applied in three broad areas – prevention and deterrence, detection, and response and recovery. Again, these responsibilities must be translated into products, procedures and products. These concerns and their countermeasures are synopsized in Table 5. The specific method used to address each concern is again a decision left to the organization and its engineers. The solution chosen is often based on non-technical rationale to include budgetary concerns, return on investment models, risk assessments, personal experience with a product, salesperson emphatic assertions, and recommendations from others. Better approaches might include reviewing evaluated product lists and internal testing of products in the operational environment if possible.
### Table 5

**Prevention, Detection, and Response/Recovery Strategy Examples**

<table>
<thead>
<tr>
<th>Prevention &amp; Deterrence</th>
<th>Detection</th>
<th>Response and Recovery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firewalls</td>
<td>Firewalls</td>
<td>Backup procedures</td>
</tr>
<tr>
<td>Public Key Infrastructure</td>
<td>Malicious code scanners</td>
<td>Forensic tools</td>
</tr>
<tr>
<td>Smart Cards for I&amp;A</td>
<td>Intrusion detection systems</td>
<td>Law enforcement</td>
</tr>
<tr>
<td>Encryption devices (hardware/software)</td>
<td>Content filters</td>
<td>Policy</td>
</tr>
<tr>
<td>OS behavior monitors</td>
<td>Audit log analysis</td>
<td>Procedure</td>
</tr>
<tr>
<td>Employee training</td>
<td>Employee training</td>
<td></td>
</tr>
<tr>
<td>Enforcement of security policy</td>
<td>Forensics tools</td>
<td></td>
</tr>
<tr>
<td>Use of high assurance products</td>
<td>Procedures</td>
<td></td>
</tr>
</tbody>
</table>

This section would not be complete without a brief comment on response and recovery – an often overlooked but essential security service. Without doubt, an unauthorized outsider will eventually penetrate a system regardless of its protection architecture. Recovery from this penetration and its likely consequences is a necessary planning function. This involves such mundane tasks as insuring system backups are acquired on a regular basis and stored in a location that would be safe should the system infrastructure be attacked or damaged. Response is an important consideration too. If a system is damaged maliciously and business loss occurs – what will be the organization’s response? Will legal remedies be pursued? Will employees that violate policy be terminated? Will law enforcement agents be called in? Such plans and decisions will need to be made ahead of time and invoked when the incident occurs. Thirty years ago, little response would be possible in that laws did not exist to prosecute against, forensics tools did not exist to gather evidence with, and law enforcement expertise in this highly technical area was virtually absent. Today this is not the case.

4.3 The Art of Security Engineering
As with other requirement elicitation difficulties that exist in traditional systems engineering, determining what the information security requirements of a customer are and how they can be best satisfied is left largely in the hands of the systems security engineer. In turn, this security engineer develops a security architecture that can address the needs of the customer and meet comprehensive system-level requirements. Customers and end users are, for the most part, incapable of articulating their security needs as anything more than high-level declarations. To develop a common understanding (or perhaps a common mental model) between the engineer and customer, some form of a business process review (BPR) generally occurs. This BPR involves the engineer, the end customer, and perhaps other stakeholders who work together to understand the current business process. Over the life of the BPR review, the team reaches a comprehensive understanding of what the current business processes are and how they can be changed within business constraints to improve security to some level acceptable in the organization. Henning\textsuperscript{21} makes the case that existing requirements engineering tools, such as Zachmann Business Process Models, can be augmented to allow common requirements engineering tools to collect raw input on information security data flows. Sufficiency in information security is achieved when the solution’s cost, in operational terms, does not exceed its value in terms of the protection it affords - a principle previously described by the author\textsuperscript{22, 23}. Starting with information gained from working closely with the customer (current processes, constraints, security policies, desired outcomes, etc.), the security engineer will generally conduct a risk assessment/analysis, a vulnerability analysis, propose an engineered solution, implement the solution, test, document procedures, and train the organization in new procedures.
This process, depicted in Figure 2, is cyclic and needs to be periodically repeated because the solution set tends to deteriorate over time as new vulnerabilities are discovered and promulgated and as new attack schemes are discovered and employed.

![Figure 3: Security Engineering Process View](image)

What appears to be the current pervasive view is an approach to securing systems that applies a defense in depth architecture to mitigate the risk to information assets down to a sufficient level as determined jointly by the engineer and the customer. A representative process for this approach is provided at Figure 4. In this environment, the security engineer composes various security products to define sufficient protection. In some cases this judgment proves accurate and in other cases not. In suggesting the architecture, the engineer generally has several templates of tried and proven solutions that are then adjusted to provided the best fit to the specific customer’s solution need.
Building a system to meet a security requirement is often difficult, because the problem being addressed is not static, but rather dynamic. Requirements such as providing an easy to use interface, online help facilities, or real time scheduling are static requirements. For static requirements, the technical solution can be determined when the system is built and delivered and that solution is generally viable for the life of the system. A security requirement is dynamic for several reasons. First, the security solution is dependent on several factors:

- the threat against the system,
- the likelihood of the threat being exercised,
- the state of technology available for system protection,
- the state of technology for system attack, and
- the perceived value of the enterprise's information assets

Second, a security solution, in most cases, needs to be developed to defend against the most likely threats. The security solution itself is also a dynamic factor. The threat against an enterprise can change depending upon specific, identifiable events. If the security solution proposed by an engineer is viewed as static - then the engineer must endeavor to establish a protection solution that addresses the greatest threat that can occur. If the solution is viewed as dynamic - then a range of protections can be proposed that address specific threat conditions and events leading to those conditions.
Third, there are no agreed upon or accepted information assurance measures or metrics that one can apply today to determine “how” secure a system is. We have no “composibility” metrics that help us understand the algebra of integrating several products together into a solution and how strong that solution is to a set of known attacks. We rather tend to use our base of empirical experience to suggest security solution sets that we have confidence in from experience or reputation and then we monitor the solution’s success during test attacks (aka, red team penetration testing) or actual performance. We also note that the security threat changes over time based on a number
of factors specific to each customer we work with and that change needs to be accommodated in the eventual solution that is engineered.

Earlier in this section, the point was made that new technology often comes along and that old security solutions do not apply or new security solutions need to be considered. In reality, this may not be a completely fair statement. What is really meant is the essential principals of security, those that were presented in section 3, need to be reapplied in a new domain or problem set. Recent examples of this can be seen in the emergence of wireless computing and high performance computing. In the future, quantum computing will again introduce new security challenges for the security engineer.

5. Conclusions

We have attempted in this chapter to outline advances over a period of some thirty years in the area of information assurance (or computer security). We have moved through a discussion of what the problem is, historical efforts to address it, design principles and commonly accepted engineering practices. We have also presented challenges that the modern day security engineer faces in bringing an adequate defense in depth strategy to bear in a specific environment. It seems appropriate to end this chapter with some challenges to conventional wisdom and historic practices as well as some heuristics applied by the author and others over time.

The world is changing and so is our ability to secure its automation. We have many products in the market place today, but we are also finding that the products do not keep pace with the problems needing solutions. Old models of security engineering do not always work well with today's problem sets. Much of security engineering is still
based on the experience of the engineer, risk management, and even luck. Software that we rely on and expect to work correctly often does not. The creation of correct software adhering to the best development practices seems to not be occurring and in fact, some suggest that the situation is worsening over time. One might suggest that, even with over thirty years of research and progress that we should be closer to being able to protect our systems, but in reality, it is easier today to mount a significant attack against systems than it was in years gone by. This is primarily due to automated attack scripts, the abundance of attack information available to malicious users, higher speed machines, higher speed networks, advances in parallel and distributed computation, and global interconnection.

Training and experience for employees and technical staff still goes a long way toward addressing much of the problem. Awareness programs and user training are important. Most important however, is the training of our systems administration staff - an area of increasing importance and one that is often sorely neglected. The technical talent shortage continues to grow and finding capable staff with experience is becoming much more difficult. Security engineering service providers that have managed to acquire a critical mass of these individuals are lower risk companies to the clients that they provide services to. Past history is important. Beginning in 2000, the U.S. Government initiated strong university scholarship programs designed to encourage faculty and students to study this area and to enter federal service after graduation as a partial means to address the skill set shortage. Meanwhile, both commercial and government entities must be educated on the value of their information, exposures in their networks, threats, risks and thus their need to consider security as a vital requirement within their larger networked computing systems.
Information assurance or security engineering is sometimes referred to as a "black art" or “arcane science”. A good security engineer should know and understand a good security design or implementation by intuition vice quantifiable measures. In some regards, security engineering is more closely related to the legal profession: it relies upon a common body of knowledge and “case law” precedents for given architectures. Security engineers are reluctant to be the first known implementation of a particular architecture – the penalty for being first is additional scrutiny and analysis, causing cost and schedule impacts. Given that commonly accepted information assurance metrics are not agreed upon today, much of what we do, the tools we choose, and the perimeters we employ are based on empirical measures and past experience. Listed below are some observations and heuristics that are founded on experience in practicing security engineering.

- **There are different security assurance needs for different application domains.**

  Government intelligence agencies are far more likely (for good reason) to demand evaluated products, formal verification of code, trusted development environments, and high-end encryption. They are more prone to use evaluated products and support international efforts to build more trusted products. Government agencies, not in the intelligence business, are far more likely to settle for less assurance to handle their abundance of sensitive but unclassified data. Intelligence agency system security must start with the initial design of the system and build security into the overall system itself. Others in Government applications may not need this rigor. Most, in fact, can quite easily accept "add on" products to build an acceptable trust perimeter around a vulnerable system. In this domain, evaluated products are important to the
customer, but not an overriding priority. Commercial encryption of browser quality is often acceptable here. Meanwhile, the commercial customer will almost exclusively rely on composition of commercial off the shelf products for their security. Evaluation of the product by third party laboratories is not a key factor today. Within this customer base, experience with products or protection architectures is the key to acceptance.

- **In many applications - past performance and emphatic praise DOES count.** This is particularly true with commercial clients who want to use products that have an excellent reputation and service providers whose livelihood depends upon reliable, predictable systems. If a product has performed well for other clients, is reasonably straightforward in its installation, has a good support structure from the vendor, and has proven itself over time - the customer and security engineer are both likely to favor it. This decision is often made without the benefit of formal product evaluation, trusted development environments, or code verification. There is nothing wrong with this approach in most applications today. Experience does count and is important.

- **History in the business keeps one from repeating past mistakes—even if it isn’t the lowest cost proposal.** There are many "start-up" companies that are beginning to seek business in the information assurance business area. Government agencies are sometimes bound by the proposal process and low bid selection. Selection of a security engineering capability based on price can (and has) led to disaster. Experience, past performance, company commitment to the IA business area, and permanent staff can be contractor pluses. Others may involve more risk. A software engineer, systems engineer, and security engineer are not the same skill sets.
There are frightening new programming paradigms taking hold in the dot com world (e.g., extreme programming - http://www.extremprogramming.org) that will likely have a negative impact on trusted development or even controlled development. Security starts with the coders and the code that is written. This is true whether the code is for an operating system, compiler, application layer, or any other executable. Testing, quality assurance, documentation, standards, life cycle development and other standard software engineering practices are important to the assurance that we look for during execution. Trends that produce code without such quality measures are also trading off assurance for time to market. Such practices represent a threat to the security of systems. Time to market pressures can lower software safety/trust/reliability. The consumer then becomes the testing ground for the program. Many have suggested that coding practices have become worse over time and not better. Programming practices that emphasize speed over peer reviews, documentation, testing, and formalism in development tend to result in less secure and perhaps less safe code.

Integration and system compositibility is a great challenge and is not being addressed to any great extent. What we mean by this is that the ability to add on products to a system and know what results is still a black art. In part, this stems from the complexity of systems and their emergent properties. It is entirely possible to install several products that individually each provide some security features/protections, yet the combination of products results in system failure. So systems must be viewed as a whole, and not just considered piecemeal. It is also possible that individual products have data that if combined with other data would
signal an attack or penetration - but there exists no framework from which products can communicate with each other. In the network management area, such products do exist. We need them in the security area too. We also need these strength measures for product compositions to support business case analysis in industry when dollar costs must be weighed against risk mitigated. The entire area of metrics and measures in information assurance is an interesting one and largely unsolved. The interested reader is invited to review the proceedings of the Workshop on Information-Security-System Rating and Ranking (ISSRR)¹ available on line at http://www.acsac.org/measurement/.

Thirty years is a long time in the technical world and progress should be expected over such a time frame. In the area of information assurance, this has been the case. The problem areas defined by the Defense Science Board and discussed early in this chapter still exist. Our understanding of this problem set and the skills to address it have vastly improved. Attacks against systems have become easier over this same thirty years as the speed of computation increases, networking becomes more pervasive, and attack methods become better known and readily accessible to those that would use them against systems.

The information assurance area of research still requires greater attention and far more research to address future needs. Coupled with this is a need for software engineering to adopt more quality development practices and produce code with fewer latent errors and vulnerable components that later become exploitable. As computing becomes more ubiquitous and a part of every person’s daily routine, it will also become a
target for those that wish to damage an individual’s system or data holdings or simply commit acts of electronic vandalism.

6. Acknowledgements

The author wishes to gratefully acknowledge the assistance and support of several individuals that have been helpful in providing material and advise for this work. First, Dr. Jack Murphy, EDS US Government Solutions Chief Technology Officer located in Herndon, Virginia, who provided the template graphics used in Figures 1 and 2. Dr. Murphy has always been willing to help and lend advice and EDS has always allowed him to so in support of outside activities. Similarly, my good friends Ms. Ronda Henning and Dr. Kevin Fox of Harris Corporation in Melbourne Florida have contributed to this work in many ways by offering their comments and by working with the author on several related efforts. Ms. Henning and Dr. Fox are both practicing security engineers with many years of experience in the field. Many of the lessons learned that were presented in the concluding remarks came from their experience. I also wish to thank the editor of “Advances in Computers”, Dr. Marvin V. Zelkowitz for inviting me to author this chapter and for the collaboration we have had together for two years while developing a Center for Empirically Based Software Engineering – a collaborative effort between the University of Southern California, the University of Maryland, Mississippi State University, and the University of Nebraska-Lincoln. This effort was partially supported by the National Science Foundation Grant number CCR-0085749
7. References


