

Evaluation Criteria for Trusted Systems

Roger R. Schell and Donald L. Brinkley

The Trusted Computer System Evaluation Criteria (TCSEC) provide the basis for evaluating the effectiveness of security controls built into computer systems. This essay summarizes the definition and requirements of the TCSEC used to classify systems into seven hierarchical levels of enhanced security protection. This essay also reviews the history, technical foundations, and basic security requirements of the TCSEC. For a number of years, these criteria have been used in specifying security requirements during acquisition of products and systems, guiding the design and development of trusted systems, and evaluating systems used to process sensitive information.

Most of us are aware that the problem of evaluating the security of computer systems has been with us for a long time. Essay 1 described the ever-growing need for building security into computer systems, wherever there is a need to share resources or information. Essay 2 described the technology that we now may bring to bear to counter threats to computer security. However, even though the available technology may be used to provide computer security, a critical question that builders, sponsors, users, and customers of a computer system must be able to answer is, "How good is its security?" In other words, "How can the security of the system be evaluated?" That is the question the techniques described in this essay are used to answer. These techniques are organized into the time-tested Trusted Computer System Evaluation Criteria (TCSEC) [DOD85], which, for those not familiar with them, this essay is intended to summarize.

Background

Penetration testing was among the first attempts at evaluating the security of a computer system, but the futility of substantially relying on that

method for the evaluation became known relatively early. In the early days of computer security, advocates of secure systems tried to follow a path of searching for ways to penetrate the systems' controls, often using malicious software in the same way a potential attacker could launch a probing, penetration, or subversion attack, as described in Essay 1. Their plan was that, failing to penetrate, they could plausibly argue that there was no way to penetrate since no way was known (to them). In this scenario, if a security hole is found, it can first be patched before the argument for security is made.

Obviously, however, as described in Essay 2, this argument suffers from both theoretical and practical difficulties. One presumes that one could test all possible programs to find any that led to a security penetration. If possible, this method of exhaustion would be effective, but it is far beyond the realm of feasibility. For any real computer, it would take so long that before the evaluation was finished, the sun would literally have burned out! Thus, any evaluation conducted by exhaustion must be so incomplete as to be ludicrous.

The lesson that was learned is that a test can demonstrate the presence of an error but, short of exhaustion of every possible combination of inputs, it cannot demonstrate the absence of errors. Practically speaking, the effort spent in penetrate and patch techniques yields poor marginal return in terms of security [SCHE79].

In 1969 and 1970, a distinguished panel headed by Dr. Willis Ware of the Rand Corporation developed and published a seminal report [WARE70] concluding that it is very difficult to evaluate a system and determine whether it provides adequate security. Essay 2 described the major findings of a somewhat later Air Force panel headed by E.L. Glaser run under a contract with James P. Anderson (both members of the Ware panel). The Air Force panel published a report [ANDE72] that identified research aimed at solutions to some of the problems identified by the Ware panel.

In response to the recommendations of the Air Force panel, during the next few years there was substantial research and development in an attempt to devise a method of reliable security evaluation. In 1977, the Department of Defense (DoD) Computer Security Initiative began, and an effort was made to consolidate the R&D gains. In 1981, the DoD Computer Security Center was formed to expand DoD efforts to evaluate computer security. By 1983, the DoD had completed about six years of work with the National Bureau of Standards (NBS) (since renamed the National Institute of Standards and Technology, or NIST) specifically focused on building, evaluating, and auditing secure computer systems. In that year, the DoD Computer Security Center published the Trusted Computer System Evaluation Criteria (TCSEC) [DCSC83]. It was published only after worked examples of meeting each requirement existed and after being "subjected to much peer review and constructive techni-

cal criticism from the DoD, industrial research and development organizations, universities, and computer manufacturers” [DCSC83].

The center’s charter was soon broadened, and the National Computer Security Center (NCSC) was established in 1985. The NCSC oversaw minor changes to the TCSEC, and it was established as a DoD standard in 1985 [DOD85]. The evaluation criteria presented in the TCSEC and overviewed in this essay deal with trusted computer systems, or computer systems that contain a trusted computing base (TCB), as introduced and described in Essay 2.

In the time since its publication, the TCSEC has been used by the DoD and other organizations in the US for security evaluations. However, some western European nations have in the meantime begun developing different criteria, the ITSEC [ITSE91]. In 1990, a group of four nations — France, Germany, the Netherlands, and the United Kingdom — published the first draft. These draft criteria, while substantially influenced by the TCSEC, were significantly different in a number of areas [BRAN91]. It is too early to tell how usable the new criteria will be for the purposes for which they are intended, since the development of successful security evaluation criteria takes many years.

Finally, in 1990 and 1991 a joint project to produce new “Federal Criteria” began within the US government at the National Institute of Standards and Technology (NIST) and the National Security Agency (NSA). The apparent goal of this multiyear criteria-development project is to eventually replace the TCSEC with a Federal Information Processing Standard (FIPS). The following paragraphs present some of the reasons why the development of practical and effective criteria is such a difficult task.

The criteria defined in the TCSEC have survived the test of time and have been used repeatedly and successfully. These criteria were devised to apply to evaluation of computer products and systems. They are applicable to general-purpose computer systems, as well as to special-purpose application systems. In characterizing the composition of the TCB, the Department of Defense standard [DOD85] says,

For general-purpose systems, the TCB will include key elements of the operating system and may include all of the operating system. For embedded systems, the security policy may deal with objects in a way that is meaningful at the application level rather than at the operating system level. Thus, the protection policy may be enforced in the application software rather than in the underlying operating system.

The TCSEC is therefore based on a set of technical principles which are broadly applicable. Since the TCSEC was published, the NCSC has developed a number of interpretations and guidelines that assist in apply-

ing the principles of the TCSEC to networks [NCSC87a] and to database management systems and other applications [NCSC91].

The technically sound foundation of the TCSEC enables it to be used by a third party to accurately and objectively measure security trustworthiness of a system. It does this in a manner which does not require the highly skilled and experienced scientists who wrote the criteria to apply them, yet which allows those less experienced who apply the criteria to come to the same conclusion as would those who wrote them. This was a key goal in the development of the TCSEC — those who apply the criteria should not be simultaneously writing and interpreting them. That situation only leads to inequities in applying the criteria as interpretations differ between evaluations. Its practicality for objective use makes the TCSEC an important technological contribution.

Summarizing that contribution is the purpose of the remainder of this essay. Note that the requirements contained within the TCSEC are simply called the *criteria* in the remainder of this essay. Note also that this essay uses a number of terms of art from the TCSEC that are only briefly defined here and in the TCSEC. The reader is encouraged to refer to Essay 2 for a more detailed discussion of the important terms. For more information about any of the example or candidate evaluated trusted systems discussed in this essay, the reader is encouraged to contact the vendor — or the National Computer Security Center at Ft. Meade, Md., for more information about how to contact the vendor directly. For further reading, Gasser [GASS88] covers most of the technical concepts from the criteria in greater detail than presented here, though without much discussion of the evaluation ratings.

Technical foundations

The foundations for the criteria are the notions that revolve around the reference monitor concept, introduced in Essay 2. Recall from Essay 2 that the primary motivation for the development of the reference monitor concept was the potential presence of malicious software, as discussed in Essay 1. The following discussion summarizes some of the pertinent concepts from Essay 2, and, along with the next section, provides a framework for the discussion of the criteria themselves.

The notions fundamental to the criteria are as follows: One should know what it is one wants to protect (that is, one should have a security policy that defines what we mean by secure). Also, one should have some mechanism, called a reference monitor, for determining every time there is an attempt by some subject (some user or surrogate for that user) to reference an object (the information), and that mechanism should validate each reference. For this arrangement to be truly secure, we must have some model of our security policy. A system having a security policy that says a subject can access an object only on Thursdays may

be secure with regard to that security policy, but not secure with regard to one that says the subject needs a Secret clearance. So we require some security policy model that defines what kind of system protection we want.

In addition, the idea of a reference monitor has with it three design requirements that are reflected in the criteria. The first of these is that if we are going to have a protection mechanism, that mechanism must be tamperproof and must be able to protect itself against invalidation. One technique that has been demonstrated in some of the penetration testing efforts is after penetrating a system, putting in some form of trap door — in other words, tampering with the mechanism itself so that later the attacker can regain control over the system. We termed this “subversion of security mechanism” in Essay 1.

Second, it is also necessary, of course, to ensure that the mechanism is always invoked. In efforts to provide security retrofits, a supposedly secure appliqué was put on top of a system, and the user was told that every time he was going to process sensitive information, he would use this appliqué. The problem was that there were many ways to get at the information without invoking the appliqué mechanism.

A final design requirement is that the mechanism should be subject to analysis and test. This turned out to be a rather difficult requirement and is one that led to the evolution culminating in the criteria. In implementation, this reference monitor mechanism will be some relatively small portion of the hardware and software. If this portion of the system meets these three design requirements, we refer to it as a security kernel.

Basic requirements

Why do we have this set of criteria? Why not some other set? The answer is that the criteria developers looked at the basic requirements that they were trying to address with a trusted computer system, and from these derived the criteria that would be sufficient to allow us to judge how well the system meets those requirements, even under the threat of malicious software. The following summarizes the requirements considered basic to the criteria. See Essay 2 for further discussion of these concepts.

The notion of controlled sharing in computer systems implies that it is possible to define what controls on the sharing of information are desired. This definition constitutes the security policy, as mentioned above. Fortunately, it is understood that all practical security policies for reading and/or modifying information (that is, for access control) can be grouped into two common classes: discretionary and nondiscretionary [SALT75] (also called mandatory). These provide a powerful and flexible tool for the design of secure computer systems [BELL91].

One type of security policy is a mandatory security policy, in the sense that an individual who, for example, has access to Secret information is (if he is going to follow the security policy) required to ascertain that somebody else has a Secret clearance before allowing him to have access to that information. Even if he may need the information to do his job, he is not to be granted access to it until he has a clearance. This mandatory restriction on the security must be similarly enforced in the computer system.

Another subtle problem in enforcing a mandatory security policy is uncontrolled downgrading. This perhaps is most easily illustrated with Trojan horses, a form of malicious software described in Essay 1. If, for example, a system has an editor that edits a Secret file, and in the course of editing that file, the Trojan horse in the editor can make a copy for an unauthorized individual in a new nominally Unclassified file, that individual can retrieve the new Unclassified file at a later time. The problem of uncontrolled downgrading really is unique to the nature of computers. Independently of computers, we trust individuals who have knowledge of classified information to exercise the good judgment not to communicate that information to unauthorized individuals, even though they may be communicating unclassified information to those same individuals. However, a computer is unable to exercise judgment; a computer only “does what it is told” by its software (which may be malicious). To prevent uncontrolled downgrading, we need assurance that the computer enforces the mandatory security policy.

One of the mandatory security policy requirements is marking the sensitivity of information. For example, corporations depend very heavily on such markings for protection of their proprietary information. Documents are commonly marked at the top and bottom of each page to reflect the sensitivity of the information contained in them. In computer systems, information is implicitly homogeneous with regard to markings. One cannot pick up a magnetic tape and tell what sensitivity the bits are. Some sort of sensitivity marking (or labeling) system is needed internal to the computer itself.

In addition to the mandatory security policy, we have a class of requirements that has been called the discretionary security policy. This class is discretionary in the sense that an individual who has control of sensitive information can exercise his discretion in determining if someone else has the need-to-know for that information (if he has a clearance). Then, and only then, is it released to him. The requirement to enforce need-to-know is closer to the historical notions of controls in a computer system where the users and the access that they have to the information are identified (for example, user/group/world, read/write/execute).

To implement these requirements, one needs to know who has access to the system; both in the world of documents and in the automated

world, there is a need for individual accountability. There should be a record of what action an individual who processes or otherwise accesses sensitive information has taken with regard to that access. For example, if he exercises his authority to downgrade information after determining it is no longer classified, that should be recorded, so that if this judgment is called into question, it can be determined who in fact made that decision.

Finally, for the security controls to be effective, in a computer system or otherwise, there must be continuous protection of those things responsible for the security. In other words, we want to control changes to the protection mechanisms. Since we are depending on the hardware and software in the trusted system to assure the protection, we must have some way to be sure that what we have evaluated is in fact the same hardware and software that are actually executing at any point in time. The requirement for continuous protection provides increased assurance that the protection mechanisms are in place as expected and that they are not subverted through malicious software. This last point is particularly noteworthy, since even experts in the field of computer security sometimes apparently forget that increased assurance is primarily aimed at protection from malicious software. The TCB is not just “built using acceptable systems engineering practices (in order to minimize errors)” [CHOK92].

Criteria overview

In this section, the criteria defined in the TCSEC are overviewed. The criteria are structured into seven distinct “evaluation classes.” These represent progressive increments in the confidence one may have in the security enforcement, and each increment is intended to reduce the risk taken by using that class of system to protect sensitive information. These increments are intended to be cumulative in the sense that each includes all the requirements of the previous. Between each distinct class and the next there is a significant jump in the capabilities provided (and in the difficulties in providing these capabilities). Furthermore, the classes form a natural evolution path such that, by choosing one of these classes, one is not precluded from going further. (A detailed discussion of and limitations on the extensibility of particular architectures in providing the basis for satisfying the requirements of progressively “higher” evaluation classes are presented elsewhere [SCHA84, SHOC87].)

The fact that the criteria scale upward in a manner that naturally extends the requirements of one class to the next higher class allows one to more fully understand the applicable principles by looking at the “highest” classes. Similarly, after understanding the principles applied in the highest classes, it is much easier to understand what is missing in

the “lower” classes. If principles had not been applied in the formulation of this set of criteria, it would have been merely a hodgepodge of ad hoc requirements. The clear correspondence between the highest classes (closer to ideal) and the lowest classes allows easier judgment of whether a system meets any class of the criteria for satisfying the security requirements in a particular application environment.

The criteria are applied on a system basis. The seven classes are divided into four divisions, termed D, C, B, and A, that are independent of any specific applications. These divisions apply to general-purpose operating systems with their hardware, or to special-purpose operating systems and application combinations.

There are seven classes, ranging from class D to class A1, and we will address them individually. They are summarized below:

Overview of Evaluation Classes

Minimal: Division D

Class D: Minimal Protection

Discretionary: Division C

Class C1: Discretionary Security

Class C2: Controlled Access

Mandatory: Division B

Class B1: Labeled Security

Class B2: Structured

Class B3: Security Domains

Verified: Division A

Class A1: Verified Design

Division D. Division D has only one class. Evaluation class D, called Minimal Protection, is easy to address. It means that the system meeting

this class fails to meet the requirements of any higher class and cannot even be expected to protect against human error.

Division C. Division C, the Discretionary Protection Division, has two classes. Systems meeting a class in this division provide some confidence that the hardware and software controls are enforcing a discretionary security policy. This means that one has some confidence that there is some protection against the class of computer misuse techniques described in Essay 1 as human error. A system meeting class C2 also provides some protection against and detection of user abuse of authority and direct probing — additional classes of computer misuse techniques described in Essay 1. This division is most generally intended to address protection against actions of users and nonusers who are not using malicious software.

Class C1. The lower class in this division is C1 (Discretionary Security Protection). This is the first point at which one may have some confidence that a system meeting this class is implementing a discretionary security policy. From the reference monitor concept, we note that there has to be isolation, that is, self-protection. User identification/authentication must be provided since there is sensitive information to be protected. There is some understandable definition of the controls provided so that at any point one can say that these are the people who will be allowed to access this information and no others.

The establishment of confidence that one has met this level (that is, assurance is primarily based on functional testing. There is some nominal set of controls that has been advertised; one carefully tests these controls to demonstrate that they function as advertised.

Systems meeting this class therefore provide some protection against human error, but they are not much help in preventing or even detecting user abuse of authority and direct probing. This is because there is no required capability to audit user actions and there may be relatively weak access control over resources.

Examples of candidates for this class could be nearly any of the major commercial systems offered today. To date, product vendors have shown little interest in having their systems evaluated with this rating as the objective. This is probably because the class C1 rating is of little market advantage against an unrated or a class D rated system; it is perceived as offering only nominal security.

Class C2. The second class in this division, C2 (Controlled Access Protection), is one in which the resources are more heavily encapsulated, at least for a subset of the resources. The controls may not be applied to all the objects in the system; they may be principally applied to the ones that one is trying to protect and has a direct interest in. These would

include objects like individual files or in some cases particular devices. In addition, in this class explicit auditing requirements are met. Not only has the individual identified and authenticated himself prior to using this system, but there can be a record of what that individual has done. There is probably an inverse relationship between the amount of audit information available and the amount of understanding people have about what is going on. So there has to be a selective way of recording the accesses that occur. In addition to having that selective recording, of course, there must be tools for examining the audit record.

Systems meeting this class therefore provide some protection against human error and some prevention and detection of user abuse of authority and direct probing.

Examples of systems that meet the requirements of this class are the operating systems for most of the major vendors' computers, including IBM, Digital Equipment Corp., Data General, Unisys, Control Data Corp., Prime Computer, Hewlett-Packard, Encore Computer Corp., and Wang Labs, as well as add-on systems for IBM computers by Computer Associates. Several of these are security-enhanced Unix look-alike operating systems. These provide the need-to-know controls that are required for a single-level mode of operation. Division C assumes that there is an implicit single mandatory sensitivity level for all the information.

Division B. Next, we move into the Mandatory Protection Division, division B, which has three classes. The hardware and software controls of systems meeting a class in this division enforce a mandatory as well as a discretionary security policy. This means that one has some confidence that there is some protection against the classes of computer misuse techniques addressed in class C2 (human error, user abuse of authority, and direct probing), as well as some protection against probing with malicious software. Systems meeting class B2 also offer some protection against direct penetration, while systems meeting class B3 provide some protection against subversion of security mechanism. This division is most generally intended to address protection against malicious software in the applications outside the TCB.

Class B1. The first class, B1, is called Labeled Security Protection. For this class, there is an explicit security policy model for the mandatory and discretionary security policies that are to be enforced. This class has been described as "C2 with labels," referring to the lack of real improvement over class C2 in assurance, coupled with the need for labels in order to enforce a mandatory security policy. For example, in the case of the usual classified processing in the Department of Defense, the security policy model for the mandatory security policy is quite straightforward. It says that if one is going to access information with a given sensitivity level, say Secret, one must have a Secret clearance. If one has access to Secret

information, one is not allowed to downgrade that information unless one is explicitly authorized to downgrade.

The requirement is that the mandatory security policy be applied to a defined subset of the subjects and objects in the system, including some of the storage objects in the system (the things that actually store the information). It is also required that there be internal labels for the subjects and objects. This is not restricted to just DoD classified processing. Those labels are to be parametric, so that in one installation the print-out might come out saying "Personnel Sensitive," whereas in another installation, identically the same system with a different set of parameters might print "Secret" or "Top Secret" at the top and bottom of the page. The parametric nature of the labels is quite important if the systems are to have broad application.

Because there has to be a security administrator with this class of system (to administer the labels, for example, on devices, and register user clearances), there must be a manual that describes how to use this system in a given installation. One of the emphases is on having suitable documentation, so the user can interface with requirements of the external controls (such as human-readable labels on output), as well as with the controls internal to the hardware and software.

By virtue of the application of a mandatory security policy on a subset of the subjects and objects in the system, one has some confidence that there is some additional protection against the classes of computer misuse techniques addressed by class C2, as well as some protection against subversion with malicious software. However, since there is really no more assurance than with class C2, there is no real protection against direct penetration or subversion of security mechanism.

Examples of systems that meet the requirements of this class are operating systems for a few of the major vendors' computers, including IBM, Unisys, and AT&T, as well as for the Apple Macintosh (by SecureWare).

Class B2. For the next class of the division, class B2 (called Structured Protection), the emphasis shifts more to support for actually evaluating a system's security. A system claimed to meet the requirements of this class must be built in a way that allows an objective assessment of whether it actually meets them. For this, the internal structure of the system must be evident. This is the first class where the TCB significantly implements the reference monitor concept.

In this class, there are identifiable security components in the system that are protected from the remainder of the system by what is called a security perimeter (that is, at least the portion of the system that contains the security-related components is inside the security perimeter). An attempt is made to show that the parts of the system inside the security perimeter are not harmful, even if they are not security-related. This rep-

resents a fairly distinct jump in capability from the previous classes. Thus, the tasks for the evaluators and the producers are to identify the security-related parts of the system and to show that the remainder of the system from which the security-related parts are not protected is not harmful. The importance of class B2 and higher classes, which also have a well-defined security perimeter, is particularly evident, even beyond security, when one considers the problem of system maintenance and support. If the evaluation is dependent on all the programs that are ever run on that processor for that system, any time there is a change in a program one must concern oneself with whether the evaluation is still valid. By limiting the portion of the system that is responsible for the security controls, one substantially eases that problem.

There should be distinct storage objects that can be identified. There should be, within the control mechanism itself, functionally distinct modules, designed so that the principle of least privilege is enforced, and one should be able to identify which module is providing which part of the protection. Hardware must separate the security-related modules from those that are not, and hardware mechanisms such as segmentation must be used to maintain the separation of objects on the basis of their different attributes (that is, whether they are readable or writable). The system may have components that are not security-related, but they must be separated through the use of hardware from those on which the security of the system depends. A descriptive top-level specification (DTLS) of the interface of the security-relevant portion of the system (the security perimeter) must be provided. The DTLS is not merely a set of manual pages, but rather it must be a complete and accurate specification that describes the TCB in terms of exceptions, error messages, and effects [PARN72a]. The DTLS must be provided along with a formal, proven representation of the security policy model.

In addition, this class introduces an emphasis on the problem of preventing unauthorized downgrading by covert channels through the mechanism itself leaking the information from one label to a lower label. At class B2, covert storage channels (covert channels involving the passing of information through a storage location, for example, through exceptions) should be identified. With this class, the focus is on identifying the covert channel problems that might be there and having the ability after the fact (after they may have been exploited) through audit to determine whether they may have been used for unauthorized downgrading.

The labels of the mandatory security policy are enforced for all the visible resources. These resources are not just the explicit storage objects, but also there should be labels for devices (for example, for communication lines). This includes single-level devices and multilevel devices. Whatever the resources are, if information is provided, there should be an explicit label for that information. For example, for multilevel communi-

cation lines over which packets are being communicated, labels for the packets are required. Control of unauthorized downgrading provides a substantially increased assurance that the system is in fact protecting the information, even from malicious software.

With this class, we also have increased assurance in the identification/authentication process. The need that this satisfies is illustrated by a well-known attack. The attacker will leave a terminal with a program that asks for a victim's password. The victim comes up to the terminal and logs in. The attacker's program will ask for the password, which will be provided. Thereupon, the attacker's program will simulate a crash of the system and save a copy of the victim's password for the attacker. This is a problem where there has not been a trusted path for providing the authentication. Class B2 requires a trusted path — a trusted way of making sure that when one is authenticating oneself, one is authenticating to the system and not just giving the password to some attacker.

Furthermore, there is increased concern for the continuous protection of the protection mechanism itself against illegal modification. For this class, there must be explicit tools provided for monitoring the configuration changes (that is, configuration management). The evaluators look at the builders' mechanisms for providing configuration management and also require tools for doing things like comparing this version of the system to a previous version.

In addition to the protection provided by systems meeting class B1, systems meeting this class also provide some protection against direct penetration but lack protection against subversion of security mechanism. Even with the increased assurance of class B2, penetration is still a distinct problem, and subversion is not really addressed. It is fortunate that we still have two classes to discuss.

Examples of systems that meet the requirements of this class are the Multics operating system developed by Honeywell, the trusted Xenix operating system developed by IBM and Trusted Information Systems, and VSLAN, a local area network developed by Verdex.

Class B3. Class B3, called Security Domains, is the final class in division B, and it addresses the remaining evaluation difficulty in this division. In class B2, there were still a lot of things to look at inside the security perimeter that really did not have anything to do with security. For example, linking mechanisms and file systems can be quite complex but for convenience may well have been put along with the basic security controls in the heart of the system. At the class B3 level, however, the intention is to have a simple, central encapsulation mechanism that separates those portions of the system that are security-related from those that may provide some necessary and common services to the users but really do not relate to security. This requires a layered set of abstract machines and an ability to separate those layers into distinct

protection domains. The paradigm here is one of extending the hardware — a well-specified, stable, unchanging machine — upward to higher layers of the system.

In class B3, it should be possible to remove from inside the security perimeter all parts of the system that are not really security-relevant. This last point is the requirement for minimization of the complexity of the portion of the system inside the security perimeter. As a practical matter, minimization forces a general-purpose system to implement at least three protection domains: one for the portion of the system inside the security perimeter (the TCB), one for the operating system, and one for the application. However, some special-purpose or embedded systems may implement the application and the operating system in the same domain.

At the class B3 level, there is a highly structured implementation of the design. There is not just a set of functional capabilities that one can add, but at this point, one has to build the system in a way that is subject to the evaluation for class B3. This evaluation process is more akin to the notion of quality assurance — of looking at how the system is being developed — than it is to that of testing a black box after the fact. Class B3 requires an ongoing effort during development to make the design and implementation understandable and inspectable to later achieve a meaningful evaluation. The motivation for this is the third reference monitor design requirement described earlier — to ensure that the implementation of the reference monitor can be subjected to analysis and structured testing in a manner to assure completeness.

A system that meets the class B3 requirements contains a security kernel, which is the hardware and software that implement the reference monitor concept. It is this reference monitor implementation that realizes the abstractions of subjects and objects out of the physical hardware resources. Beneath the set of security kernel interface functions is a set of computer code, databases, and hardware structures. Above the security kernel, one can expect to find a set of additional security-relevant TCB code and databases that implement a discretionary security policy and support the identification/authentication and audit functions. The TCB is structured in a series of layers, each performing a distinct set of services either for processes outside the security-relevant portion of the system or for other security-relevant layers, and each depending only on the services of lower layers. A layered system provides an opportunity to develop an informal proof sketch of each layer's sufficiency to meet the requirements of the next higher layer, based on an implementation specification for each layer. The proof sketches for succeeding higher layers of the system, if performed, would provide a clear correspondence of the layers in the design to the DTLs and hence to the formal security policy model. The correspondence of the layered implementation to the security policy model is also facilitated by breaking the implementation design into modules with a clean, clear abstraction that allows for informa-

tion hiding within modules. Note that since module, layering, information hiding, and abstraction are all used in the TCSEC as terms of art, they are not fully defined there. However, these terms are well-described in the computer science literature (for example, in the early groundbreaking software engineering work of Parnas [PARN72a, b]).

Therefore, class B3 is the lowest class at which systems that meet it have at all addressed subversion of security mechanism. In a class B3 system, one can have some assurance in protection from subversion, in addition to significantly enhanced protection against direct penetration beyond that provided by a system meeting class B2.

A commercial class B3 system is the XTS-200/STOP from Honeywell Federal Systems, Inc. The XTS-200 is a multiprocessing superminicomputer, capable of supporting up to four independent processors. STOP is its multitasking system, which supports much of the Unix interface for application software.

Division A. Division A is distinguished by additional requirements substantially dealing with the problems of subversion of security mechanism. To gain increased assurance beyond division B, one needs more structured verification support tools. In division A, which includes only a single class, formal and informal analyses are required to support claims of the correspondence of the implementation to the interface specification and hence to the security policy model. This enhanced assurance means greater protection against subversion, as well as against direct penetration and the other classes of computer misuse techniques discussed in Essay 1.

Class A1. For class A1 (Verified Design), which is at the limit of the state of the art, mathematical tools are applied that use formal security policy models with explicit security theorems. A formal top-level specification (FTLS) of the TCB interface (the security perimeter) is required. This form of specification is expressed in a formal language capable of being analyzed by a computer program and of supporting a formal proof that the specifications conform to the requirements of the formal security policy model. The FTLS is analyzed in a systematic way, in order to gain a high degree of confidence that if that specification is correctly implemented, the security perimeter that is expected to have been provided will in fact have been provided. This FTLS is used for the analysis of covert storage channels that is required to use formal tools.

In addition to this analysis, it must be assured that the implementation corresponds to the specification. With the current state of the art, this will not be a wrapped-up mathematical proof that the system is secure since, in general, correctness proofs at the source code level are not practical. However, increased assurance over that of a class B3 system is afforded by an analysis showing the correspondence of the source

code of the TCB to the FTLS, supplementing the layer-oriented analysis required for classes B3 and A1. The formal analysis is assisted by tools that can provide evidence in various degrees of formality, both at the specification level and at the level of the implementation correspondence.

In addition, configuration management becomes of increasing interest as these formal methods are introduced. Class A1 requires control not only of the hardware/software mechanism, but also of the specifications. Control must exist throughout the life cycle — during the system's design, development, production, and distribution. That says there is concern for the trusted distribution of the system. This trusted distribution requirement reflects that as we move to class A1, we expect to have increased dependence on the system. We expect a decrease in the risk from using it, and therefore we are more concerned about its vulnerability to subversion during the distribution process (for example, by replacement of part of the TCB with malicious software from an attacker).

The result of the enhanced assurance with class A1 is, as noted above, greater protection against subversion of security mechanism, as well as against direct penetration and the other classes of computer misuse techniques discussed in Essay 1.

An example of a system that was evaluated as meeting the class A1 requirements in 1985 is the Honeywell SCOMP minicomputer. We have already discussed its successor, the XTS-200, which is a class B3 system.

The Boeing Multilevel Secure Local Area Network Server System meets a subset of the class A1 requirements that are particularly applicable to networks. This subset, an evaluation of which is defined in the Trusted Network Interpretation (TNI) of the TCSEC [NCSC87a], includes support for enforcing a mandatory security policy between attached devices and identification/authentication of users of some of those devices. It supports devices such as terminals, host computers, serial devices, video devices, and stream devices.

The Gemini Trusted Network Processor (GTNP) from Gemini Computers is another candidate to meet the mandatory security policy requirements of class A1, as interpreted in the TNI. The GTNP product consists of security software that operates on a hardware base with up to eight processors (Intel iAPX 80286 or 80386) in hard-disk-based, floppy-disk-based, and RAM-based models. These systems include Gemini-specific device interfaces as necessary to support a particular configuration. The system takes an open-architecture approach in which applications and protocols may be developed to run on top of the base provided by the GTNP to support specific network requirements or network component compositions, without affecting the GTNP TCB. The GTNP provides device interfaces for several types of networks, a preallocated address (MULTIBUS I) interface, and a virtual machine monitor interface that

can be used to implement applications such as guards and trusted database management systems [IRVI91] without requiring reevaluation of the underlying product. It also provides support for encryption and cryptographic sealing of critical system structures (including labels) and user data using the Data Encryption Standard (DES).

In modern communication systems, a dedicated, RAM-based GTNP configuration is well-suited as a secure front-end or interface processor. The flexible multiprocessor architecture provides low risk and at the same time meets demanding throughput requirements such as those encountered in high-speed packet switching or end-to-end encryption control [WEIS92]. The integral DES encryption hardware also makes it attractive for guard interfaces where encrypted checksums or digital signatures are used to establish the authenticity of received information. In workstation environments the disk-based configurations are well-suited for distributing the processing of sensitive information, especially when the workstation is part of a network. In addition, the high performance available from multiprocessor configurations offers unique opportunities for specialized systems such as high-capacity multiuser database management systems.

Beyond class A1. In considering the future, it is useful to note that the criteria have specifically provided for the addition of more evaluation classes in division A as the state of the art progresses. It is particularly important to note that the underlying technology provides the foundation for even higher assurance than what is realized by the current practices, that is, class A1 and lower. Thus, while class A1 provides truly outstanding security, it is reasonable to expect even greater security in the future for those applications facing extraordinarily intense and hostile threats, especially of penetration and subversion using malicious software.

For these classes, applications may have even greater dependence on the trusted system for security, so there is increased concern for the development environment (for the correctness of the tools used in the development). There is also increased concern for verifying the correctness of the TCB to a lower level (down to the source code level), including hardware and firmware. These are goals for future criteria.

Evaluation by parts. In dealing with any complex system, an attractive strategy is one of “divide and conquer.” What is desired is an approach for dividing the trusted component of the system into simpler parts, evaluating each of the parts separately, and then validating the correctness of the way the parts are composed to form a single system enforcing a globally well-defined security policy. It is obviously desirable that the validation of the correct composition of evaluated parts be a relatively simple task. The simplicity of the reference monitor concept

lends itself well to a divide-and-conquer evaluation strategy. In this section, two distinct strategies will be briefly overviewed. These ideas are an extension of the principles first described elsewhere [SCHA84], with the benefit of several additional years of experience and thought about how a complex TCB might be composed from a collection of simpler TCBs residing in individual protection domains.

The first strategy, the “partition evaluation” of a trusted distributed system or network, depends on the notion that a complex system may be decomposed into independent, loosely coupled, intercommunicating processing components. This strategy is most suitable for trusted systems that have a complex physical architecture (for example, distributed systems or networks). The partition evaluation strategy [SCHE85] is the basis for Appendixes A and B of the TNI [NCSC87a], which provides interpretations for the application of the TCSEC to networks.

The second strategy, the “incremental evaluation” or “subset subset evaluation” of a TCB, was used by Gemini Computers to internally structure the TCB of its GTNP and was first publicly presented by Shockley and Schell [SCHOC87]. It builds on the idea that a complex TCB may be divided into simpler TCB subsets, each of which enforces a subset of the global security policy. The effect of such a security policy decomposition, when allocated to the various TCB subsets, is to allow performance of a chain of simpler evaluations (each an evaluation of a subset), leading to an overall conclusion that the composite TCB is correct. Unlike a TCB consisting of multiple partitions, incrementally evaluated TCB subsets are thought of as residing in the same tightly coupled processing component. The subset strategy is particularly well-suited for TCBs that enforce relatively complex security policies over virtual objects which are very different from the physical storage objects provided by the processor hardware. For that reason, the TCB subset strategy is particularly appropriate for the application of the TCSEC to trusted database management systems. It is a key concept in the Trusted Database Interpretation (TDI) of the TCSEC [NCSC91], which provides interpretations for the application of the TCSEC to database management systems and other complex applications.

The partition and subset evaluation strategies are compatible and may be combined in various ways for TCBs that are complex in both architecture and security policy. This has been proposed, for example, for an embedded secure database management system [IRVI91].

As we have noted, there are fundamental differences between the degree of protection from malicious software offered by the enforcement of a mandatory security policy and that offered by a discretionary security policy. This difference has led to the development of a useful technique for enhancing assurance in parts of the TCB where it matters most. The technique is balanced assurance.

Using partitions or subsets, one of two positions for assurance in a TCB can be adopted. The more conservative approach holds that all assurance requirements for a particular evaluation class must be applied uniformly to the entire TCB. This is termed “uniform assurance.” The less conservative position requires the application of assurances to partitions or subsets only where the assurances are relevant to the security policy enforced by the partition or subset and provide a genuine increase in the credibility of security policy enforcement. This approach is called balanced assurance [LUNT88a].

Balanced assurance is endorsed by the TNI’s Appendix A (though not by name) through its stipulation of a maximum class of C2 (with respect to assurance) for those TCB subsets not enforcing or supporting the mandatory security policy. This means that, using the TNI, one can have a class A1 network that contains partitions (for example, enforcing discretionary access control and audit) that meet only the class C2 requirements for assurance. Balanced assurance is an important technique for achieving near-term high assurance where it counts in complex systems, but it is still controversial since it is a less conservative approach.

Conclusions

The criteria summarized in this essay give us the technology to build and objectively evaluate trusted computer systems. However, the techniques of applying the criteria are not the most difficult challenges that we face in maintaining a technology base on which secure systems can be efficiently developed and used. Managers of information systems must also meet their responsibilities to maintain a constant commitment to protect the information with which they have been entrusted. Moreover, trusted product developers must continue to provide the building blocks that enable trusted computer systems to be efficiently developed for specific applications.

Sound criteria. The practicality of applying the principles of the TCSEC and its interpretations, which were formulated based on both business and technical considerations, is demonstrated by the fact that TCSEC’s development was based on worked examples. In fact, before there was a TCSEC, there were worked examples of the use of each of its principles. For example, TCB isolation in the first system to achieve a class A1 evaluation rating — the SCOMP, whose development was begun significantly prior to the completion of the TCSEC — was provided by a hardware implementation of a ring mechanism that was developed in the late 1960s and early 1970s for the Multics system [FRAI83, SCHR72]. The Blacker system, which was developed during the formulation of the TNI, confirmed the practicality of notions such as TCB partitions [WEIS92]. Similarly, the notion of TCB subsets [SHOC87] was

incorporated into the TDI only after experience with it in the design of the SeaView multilevel database management system [LUNT88].

It was considered quite important during the formulation of the TCSEC to require, to the extent possible, only the minimum additional documentation for evaluation evidence beyond that produced already by a quality-conscious developer. The majority of the evidence is intended to be exactly that which the developer needed to design, build, and maintain the system. This goal of having most of the evaluation evidence produced as a by-product of the trusted system development is based on the principle that the evidence is produced for the developer — not for the evaluators. This goal is revealed in the fact that the TCSEC does not enumerate items to be “delivered” to the evaluators and does not impose requirements for “presentation” for the documents and evidence.

Another important goal in the formulation of the TCSEC was to support the procurement and approval processes by minimizing the number of discrete evaluation ratings — by providing only the ratings between which there was a clear difference in the protection offered by a system. Further, in the discrete evaluation classes, there is a coherence between features and assurance, so that they are combined in an intelligent fashion. The idea is that there is no need to encourage or even to have systems which possess higher assurance, without corresponding gains in the features. For example, there is no class C3 — offering higher assurance of a discretionary security policy. Since a discretionary security policy is fundamentally incapable of protecting against malicious software, there is no practical reason to have higher assurance of its enforcement beyond that offered by a class C2 system. For this reason, the TCSEC provides an enumeration of all evaluation classes; as stated above, there are only seven. The principle here is that the scientists who developed the criteria are in a much better position than are others, by virtue of their understanding of the principles, to determine the features and assurances that work together to counter specific threats.

As noted in the sections on technical foundations and basic requirements, the technical soundness of the TCSEC comes from its basis in valid scientific underpinnings. Fundamental to the TCSEC is the reference monitor concept, which implies the notion of having a security policy that possesses properties such that whether a given system enforces those properties is computable (see our statements about this in Essay 2). For this reason, the TCSEC provides criteria for the evaluation of a system with a definitive security perimeter that enforces access control policies. As stated in Essay 2, whether or not a given system meets the criteria of availability, reliability, safety, and other such properties is fundamentally noncomputable. The foundations of the reference monitor concept also imply (as described in the section on class B3) the minimization of the portion of the system inside the security perimeter, as well as

the paradigm of verification of succeeding layers of the TCB by the development of an informal proof sketch for each layer. These reference monitor principles are key to attaining high levels of assurance and provide the basis for understanding what is lost by relaxation for the lower levels of assurance.

The practicality of applying alternate criteria that are just being placed into use (for example, the ITSEC [ITSE91]), as well as those under development (for example, the Federal Criteria), has yet to be significantly tested. However, these criteria are being developed based on considerations different from the business and technical considerations discussed above for the TCSEC. (In fact, the ITSEC differs from the TCSEC on each of the considerations described above in this section!) The results may affect both their usefulness (the degree to which product vendors and system procurers are willing to subscribe to them) and their soundness (the degree to which insecure implementations could be evaluated as secure and vice versa).

The criteria have a sound technology base, broad applicability, and several years of application experience in real evaluations. As it stands, the criteria are highly objective and effective as a measure of the appropriate confidence in the internal hardware and software controls. However, there have been reports of concerted opposition that would weaken the criteria's effectiveness ever since the TCSEC was first published [SCHE84]. For example, some have raised specious arguments for consciously limiting the criteria's use to just the DoD, implying that these criteria are applicable only to the DoD. Although there is a superficial appeal to this argument, it ignores the fact that both the underlying technology and the criteria themselves are not tied to DoD-unique characteristics. This argument also ignores the growing need for trusted systems that we have addressed in Essay 1. On the contrary, Shockley [SHOC88] demonstrates that this technology can satisfy a security policy claimed to be "an accurate representation of what the business and commercial data processing community means by the term 'integrity.'" Further, Bell [BELL91] shows how it is possible to support a number of non-DoD security policies with this technology.

In addition, both inside and outside the DoD, there have been efforts to make "plausible" but in fact devastating changes to the criteria that would seriously weaken the technical foundations. It is just these foundations that make it possible to clearly determine the effectiveness of the security. For example, eliminating the requirement for a proof of the security sufficiency of the rules of the formal security policy model would make the criteria much easier to meet, but correspondingly much less useful and in fact even seriously misleading.

In view of this sort of controversy, we must recognize that as an industry, it is an essential but elusive goal to have widely accepted, meaningful criteria. However, we must also recognize that technically weak

criteria would deprive those with the critical need for trusted systems of what they need most — a tool for objectively determining the security of their systems in the face of practical attacks, including malicious software.

Responsibility for information protection. There is little in the way of laws, regulations, policy, and practice to mandate or even encourage meaningful hardware and software controls (especially at the higher assurance levels). Hence, some observe that there may be little incentive for management to buy the trusted products that are available or to implement trusted systems. It has been noted that many of the managers who are responsible for selecting the particular computer system an organization will use have little at stake in protecting the information they process. The serious potential damage is often not to their immediate group and may not even be to their parent organization — for example, where computers are used to provide service based on someone else's information. Even though an information service may be in a position to seriously damage individual privacy, the organization may feel it has little to risk itself.

To date, the response to the available trusted products has been less than overwhelming, even in the DoD, where use of a trusted system of sufficient assurance to meet the risk faced by each specific system is a part of DoD policy [DOD88b]. For example, the DoD, which sponsored the major security enhancements to the Honeywell Multics product, used this system in very few installations, even though various assessments highlighted the need for that type of security. (Perhaps as a consequence, Multics is no longer available as a standard product.) The DoD has more recently encouraged the broad use of lower assurance products (for example, through the "C2 by 92" initiative). However, it is essential that managers within the DoD begin to mandate use of trusted systems of sufficient assurance to meet the requirements of DoD Directive 5200.28 [DOD88b] (enclosure 4) if they want to prevent the moral equivalent of discussing classified information on open telephone lines.

If the appropriate trusted systems are not used in commercial as well as government installations, we are, in time, almost assured of a disastrous computer-security "Chernobyl"-like event of major proportion, in which serious damage will be done due to the continued negligence of those responsible for computerized information. Although this could well stimulate the use of trusted systems at that time, there is a high risk of overreaction in areas such as government regulation and restrictions on the use of automation. On the other hand, use of sound criteria to determine whether or not the security policy of a particular system is being enforced to an appropriate degree of assurance (that is, determining that the evaluation class met by the system satisfies the security requirements for the environment in which the system will operate) can go a long way to prevent a disaster.

Availability of trusted products. Trusted product availability at lower assurance classes is steadily improving. In addition, the criteria have been successfully used to support several assurance evaluations at class B2 or higher (including systems such as Blacker — which used “GEMSOS, an off-the-shelf kernel from Gemini Computers” [WEIS92] — and commercial products). Yet there are still only a small number of higher assurance product choices available. In addition, some of the products evaluated by the NCSC have not fared well in the marketplace. This is clearly related to the lack of overwhelming demand for those products that are available, as just addressed.

However, several vendors are seriously developing trusted systems. Also, the NCSC and NIST continue to respond to the growing need for computer security within the US. Among other things, as mentioned earlier, the NCSC manages evaluations of commercial products against the established criteria and publishes an “Evaluated Products List” so that customers can better determine what products are available.

On balance, we are persuaded that, as more trusted computer system products become available that are both practical and secure, managers will step up to their responsibility to use them to protect the information entrusted to their machines. We as computer professionals must accept our part of the responsibility for information security now and in the future. As we do, trusted systems will have an increasingly valuable place in our profession. Since it is not practical to individually evaluate the security of each system starting from basic science, the criteria are the primary tool that makes it possible for us to address our information security responsibilities effectively.

