

## Essay 14

# Security Engineering

Marshall D. Abrams, Harold J. Podell, and Daniel W. Gambel

---

This essay is concerned with trusted system integration and/or development to meet multilevel security (MLS) and operational requirements. It addresses technical issues such as how to combine products securely, TCB alternatives, and typical security engineering phases — as well as the management concerns of certification and accreditation.

This essay addresses the integration of multilevel security (MLS) technology into the concept definition, acquisition, design, product selection, and MLS integration phases of an operational system. Trade-off analysis is required among factors such as technical risk, security risk, cost, and satisfaction of operational requirements. The essay is divided into four phases:

1. In the requirements phase, we discuss policy determination, the need to identify trust requirements, application of user and mission requirements, use and development of the security Concept of Operations (ConOps), applications for scenarios, and selection of the correct version of security policy.
2. During the design phase, we discuss how to apply design guidance and regulations, and consider the advisability of including certification team participation in design.
3. Discussion of the integration phase surfaces issues from MLS integration policy needed, how to combine products securely, determination of whether to build or buy a TCB, use and considerations of trusted and untrusted processes, considerations for porting untrusted applications to a TCB, and approaching complex systems.
4. In closing, we discuss aspects of certification and accreditation, including the role of certification and accreditation (C&A) and establishing a C&A program.

Focusing on the end objective of getting the system accredited for operation, Bauer [BAUE91] identifies the key issues of the development process to be:

- consideration of mission requirements and security requirements prior to allocating requirements to trusted mechanisms,
- trade-offs between security disciplines and between overall security versus mission requirements, and
- structure of complex integrated systems using custom developed and commercial off-the-shelf (COTS) components.

However, a system integration normally involves a contract between the integration organization and a customer. Unlike a product evaluation, which has a primary goal of evaluating a vendor's secure system, the integration is based on the process of an integrator meeting a customer's operational mission. In many cases, the security features are secondary, while assurances are relegated to a distant secondary role.

It may be true that the system cannot be used if it is not adequately secure, but it is more true that if the operational requirements are not met, the system development either will not be funded or will not be accepted for contractor payment. This difference means that the requirement definition and interaction desired, and possibly achievable in a research environment, are not possible in a system integration. The system engineering trade-offs are not under the control of a single entity. Critical trade-offs occur prior to commencement of the system development by the prospective integrator and become determining factors in the system price. The structure of the final system is a result of the proposal preparation and evaluation process, and rarely represents an optimum set of answers.

## Requirements phase

**Policy determination.** The system requirements phase includes determination of the applicable security policy. One of the first steps in security engineering is to identify the objective. As mentioned in Essay 13, there are varying statements of the security objectives in an organization, depending on an individual's responsibilities and job function. However, the TCSEC design and evaluation paradigm is predicated on the notion that "design and analysis of systems proceeds from policy to mechanisms" [TINT92].

At the general management level, the objectives of the system security policy have to be stated in generalities; while at the specific system implementation level, the objectives should be very concrete. The result will be a layered policy in which each main layer relates to a set of re-

sponsibilities. All layers must fit together in the sense that the policy described for a lower layer must be an implementation of the policy at a higher level.

The customer organization has an interest in protecting its information assets against undesirable events expressed in a high-level information security policy. This policy should address:

1. the information assets of the organization,
2. the damage that can occur to those assets, and
3. the measures that management has decided are reasonable, cost-effective, and proper to protect those assets.

Top management will probably not need to understand or care about the implementation details, as long as the information management policy is satisfied. At this high level, all policies merge into something like “Protect the organization’s information assets.” While it is easy to understand the sentiment behind such a generalization, it does not provide much guidance on whether specific actions are permitted or not. Introducing such specificity requires a significantly more detailed interpretation of the high-level policy. The specification, design, and implementation of a specific system require a different view of computing than that of the top management. But a greater knowledge of computing does not permit the creation of new policy. The AIS (Automated Information System) system policy is expected to be an implementation, at a lower level of detail, of the policy adopted by top management.

Many organizations provide very specific implementation guidance. This guidance should serve as a statement of policy that controls the solutions available to translate the top-level information security policy into a system. In general, the technical policy and solution available in a system implementation differ in substance and intent from the higher level prose abstraction. It may be impossible, or at least exceedingly difficult, to directly implement the natural language policy as rules within the computer. The security engineer is responsible for demonstrating the cost-effectiveness and correspondence between the high-level and implementation policies.

Since natural language is prone to ambiguities and multiple interpretations, information security policy is often expressed in a mathematical formalism. For more information about models, see Essay 8. Implementers should be able to rely on the formal model to resolve any ambiguities concerning the policy to be implemented.

**Identify** trust requirements. The TCSEC design philosophy is to separate trusted and untrusted code, relying on the trusted code to prevent policy violations. The trusted code constitutes the trusted computing base (TCB). But all TCBs are not created equal, and the customer’s security

engineer has to select the trustworthiness that provides sufficient countermeasures for the risk environment and simultaneously sustains the operational requirements set.

The TCSEC provides a set of countermeasures and associated assurances, as summarized in Table 1. This binding of functionality, strength of mechanism, and assurance is policy within the US DoD. Essay 7 discusses additional policy expressed in the Environments Guideline (Yellow Book) [CSC85] and DoD Directive 5200.28 [DOD88b] that relates TCSEC evaluation class to the risk environment. This foundation of policy applies only to the concepts and countermeasures associated with the confidentiality protection of classified information. In all other situations, including protecting information from unauthorized modification and assuring reliable service, the customer's security engineer must assist management in deciding what countermeasures and assurances are appropriate in terms of cost, feasibility, and inconvenience.

**User and mission requirements.** All security engineers must understand the security requirements as they relate to overall mission effectiveness. To achieve this understanding, the security engineer must develop a much broader outlook toward the system than simply security policy. Of particular concern are the constraints that can be placed in an unwelcome manner on information flow as necessitated by various combinations of countermeasures. It is rare that users see a need for sophisticated security technologies; they merely experience unwelcome constraints on their mission.

The engineering understanding of the system operation and its relationship to the operational requirements of the system are expressed in the system Concept of Operations (ConOps). This document communicates for the customer the integration of system requirements into a single view of how the resulting system will operate, given the total set of requirements. The document addresses all aspects of the operational environment, including communications, operations, data management, and applications of high-level security. However, a detailed examination of how the security mechanism operates is usually addressed in a subordinate security-specific document, the security ConOps. The development of a security ConOps and security scenarios to illustrate that unique security ConOps is important in gaining an understanding of user information security needs and the methods of addressing sensitivity of information.

**Security Concept of Operations (ConOps).** During the earliest phases of any project, mission needs are determined, alternatives explored, and cost and feasibility studies conducted to optimize the combination of requirements — and finally a high-level operational concept is explored and defined in the system-level Concept of Operations.

Once this overview of the system is expressed, an extract and amplification of the security-relevant approach is documented in the security ConOps.

**Table 1. TCSEC summary chart.**

		C1	C2	B1	B2	B3	A1
Security policy	Discretionary access control	■	■	=	=	=	=
	Object reuse		■	=	=	=	=
	Labels			■	■	=	=
	Label integrity			■	=	=	=
	Exportation of labeled info.			■	=	=	=
	Multilevel export			■	=	=	=
	Single-level export			■	=	=	=
	Human-readable output			■	=	=	=
	Mandatory access control			■	■	=	=
	Subject sensitivity labels				■	=	=
	Device labels				■	=	=
Accountability	Identification & authentication	■	■	■	=	=	=
	Audit		■	■	■	■	=
	Trusted path				■	■	=
Assurance	System architecture	■	■	■	■	■	=
	System integrity	■	=	=	=	=	=
	Security testing	■	■	■	■	■	■
	Design spec. & verification			■	■	■	■
	Covert channel analysis				■	■	■
	Trusted facility management				■	■	=
	Configuration management				■	=	■
	Trusted recovery					■	=
	Trusted distribution						■
Documentation	Security users guide	■	=	=	=	=	=
	Trusted facility manual	■	■	■	■	■	=
	Test documentation	■	=	=	■	=	■
	Design documentation	■	=	■	■	■	■

**Key**

No requirements for this class

≡ No additional requirements for this class

■ New or enhanced requirements for this class

The security ConOps emphasizes security over the other operational aspects to focus the technology on this constraint-prone aspect of a system design. Identification and agreement are required for security modes of operation, transitions between modes, and identification of

1. the sensitivity range of information,
2. the clearance range of users, and
3. security-relevant characteristics of external interfaces.

A key part of the security ConOps is a section that describes results from analysis of the anticipated flows of information:

1. where information originates,
2. where and how the information is anticipated to be received, and
3. the classification of that information.

The results of the analysis are used to identify cost-effective applications for MLS technology.

Requiring a security ConOps to be prepared and submitted by the prospective integrator during the competitive cycle of the system acquisition is an effective approach. Advantages of this approach include establishing a contract baseline, identifying applicable potential concepts for MLS technology insertion, and describing the security aspects of each competing design. If the security ConOps is to be delivered as part of the contractual deliverables, it should be initially delivered early in the design cycle in concert with preliminary design documentation.

The security ConOps should have a wide review audience, including customer and integrator management, software developers, end users, and the certification and accreditation team. Revisions to the security ConOps resulting from the design process (if procured) should present significant security issues. These issues are reviewed to provide validation and keep the review audience abreast of the evolution of the approach.

**Scenarios.** Scenarios are useful functions during several phases of the evolution of a system, particularly during:

1. the requirement definition phase,
2. the definition of demonstration or rapid prototype requirements,
3. the initial operational evaluation, and
4. technology insertion for a developed system.

One scenario composition strategy is to focus on areas likely to be affected in a substantial way by the introduction of MLS technology. By focusing on these areas, the customer's security engineers and end users could evaluate options and determine what specific MLS features and capabilities are required.

A second strategy is to use scenarios to evenly evaluate the proposed integrator's solutions when using commercial off-the-shelf technologies. Still later, scenarios are useful in developing test strategies and ensuring functional security support for all phases of the mission execution. Rapid prototyping can be a valuable method to introduce MLS concepts to the end users early in the acquisition using a site-specific application, especially as compared with abstract specification or criteria notations (for example, subjects, objects, read down, write up).

Another useful function for scenarios is to develop a range of flexibility for the User Interface Sets (UIS) [FERR91]. This conceptual approach to risk management provides the accreditor with significant insight into the real impact a user can have on the system security. Thus a rigidly controlled icon- or menu-driven interface can be used to restrict the operational user from access to capabilities (such as compilers) which may be used to explore or exploit the system security posture.

**Security policy.** A security policy specifies how to manage, protect, and distribute sensitive or critical information. Most of the time, the customer's security engineer supporting the acquisition will have to help select or identify the applicable security policy. In addition, the security engineer may be required to resolve conflicts between identified security policies from different layers of the organizational infrastructure. Most organizations are relatively hierarchical; therefore, it is logical to develop the information system security policy using a top-down hierarchical approach.

The results of the analysis can be represented using a structure that resembles a tree with corporate/national policy at the top of the tree and local policy and practices filling out the lower portions. This approach provides a "family tree" of security requirements. Some of the documents and their relationships are not necessarily common knowledge. For example, in some cases, they may have to be written. Identifying all of the documents and their relationships gives the customer's security engineering team a firm understanding of the overall security requirement. The family tree provides a common reference point for discussions with groups outside the security engineering team.

Once the customer's security engineering team has resolved the conflicts between policies and structured the system-unique policy, the customer's team should continue with the development of the eventual integrator's contractual requirement for the system undergoing acquisition. This approach is distinctly different from development of a product:

With a product, the team defining the policy is normally the team developing the solution.

In acquired systems the customer's requirement team is defining the set of constraints or policies that will be considered or established as requirements during the acquisition. It is a rare situation where a revision to the security policy can be considered by an integrator subsequent to issuing the acquisition without significant impact on the cost and schedule of the system delivery.

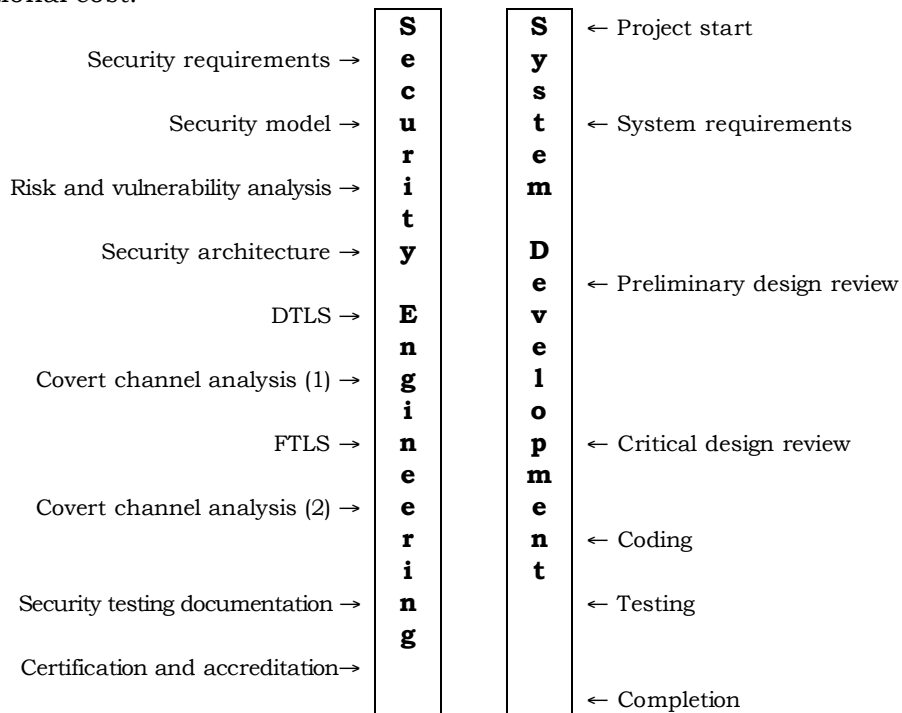
## **Design phase**

**Design guidance and regulations.** The DoD has separate sets of guidance governing the design and development of software systems and trusted systems. The system integrator is faced with the necessity to comply with the Contract Data Requirement List (CDRL) and the Statement-of-Work (SOW) for the contract. This does not necessarily result in the development of the standard TCSEC or system design documentation. In most cases, the TCSEC documentation for the products underlying the system architecture has historically been assumed to be adequate for the system itself. The result of this assumption is that the system adaptations to support the integration are poorly documented. In addition, there is little opportunity to develop documentation unique to the integrated system environment and remain within the cost and schedule originally conceived for the acquisition.

The general technical guidance for trusted systems design is contained in the TCSEC under the concept of a product evaluation; the DoD guidance for the design phase of a system development is contained in "Defense Systems Software Development," DOD-STD-2167A. [DOD88a]. Where the system integration has been structured to acquire both TCSEC-style and DOD-STD-2167A documentation, the two concepts can be reasonably synchronized. Figure 1 illustrates the time lines for the two sets of requirements and attempts to line them up one with the other.

However, an integration of commercial off-the-shelf (COTS) products cannot be guided by adherence to either set of documentation guidance. Strict adherence to the DOD-STD-2167A process would ultimately result in allocation decisions unmatched by any TCSEC-evaluated product. This approach invariably results in a custom system design rather than an integration. While DOD-STD-2167A can be subverted to use COTS products, there is significant risk of misunderstanding and conflict between the integrator and the customer organization. During the acquisition process, there is little opportunity for the customer to resolve at which point in the design process the integration may transition from the custom design philosophy to the COTS design

philosophy. For competitive reasons, there is little incentive for the integrator to examine unspecified alternatives that would result in additional cost.



**Figure 1. TCSEC and 2167A development requirements.**

In a similar manner, vendor TCSEC documentation developed for a product has no relationship to the environmental and operational concerns that must be made a portion of the integrator’s requirement set. This is not a fault of the documentation standard, the evaluation process, or the vendor. Rather, it is a situational fact. When the product vendor developed the TCSEC version of the document, the specific implementation being integrated was not foreseeable. Therefore, the documentation could not specifically address the environmental or operational needs of the specific system. In a like manner, the vendor was required to satisfy the documentation needs of the evaluation community in describing the generic application and use of the security features of the system. These requirements meant that the vendor developed the system documentation for the generic evaluation, not the operational system environment.

During system integration, it is normally necessary to modify existing products, develop “trusted processes,” and intentionally violate product policy assumptions to meet system specifications. Each of these types of integration actions creates system changes that mandate modification of the vendor’s product documentation. In most system integration environments, the TCSEC generic documentation is either invalidated or superseded. However, without adaptation for use in the customer environment, the resulting system integration would be unable to meet the customer’s real purpose for the system.

Many customers have the misunderstanding that detailed application-specific, or environmentally specific, tailored documentation is automatically provided with a system. This type of customer misinterprets the requirement for TCSEC compliance as calling out TCSEC documentation, but tailored for the system. This belief that calling out the TCSEC, or even TCSEC documentation, as a requirement will result in usable system documentation has been repeatedly disproved.

The customer organization must establish the minimum set of reasonable documents that is cost-effective, while establishing the necessary discipline for the successful system. In many ways, it is the discipline mandated by requiring the documentation rather than the document itself that causes a gain in assurance. In other cases, it is critical to advise users and administrators on how to operate the selected set of features. In all cases, the need for a given document must be viewed in terms of cost-effectiveness and applicability. The integrator will be required to deliver only that documentation which the customer has specified for purchase.

**Certification team participation in design.** The TCSEC approach was formulated under the concept that the product evaluation team would be working in concert with the vendor development team to achieve and evaluate a secure product. This team effort was to be concerned with attaining the requisite level of adequacy of the security aspects of the system. There is little incentive for the evaluation team to consider operability or performance. These operational considerations are left to the vendor to address. The vendor evaluation team is thus frequently placed in the position of accepting trade-offs, reducing operational capability simply to satisfy security evaluation team desires.

In contrast, the certification team assigned to an integration cannot ignore the full set of requirements applicable to a system acquisition. This team will be faced with relaxing some levels of security functionality or assurance to meet operational requirements. This distinctly different trade-off environment makes it critical that the certification personnel be involved in all aspects of the system design, development, and integration.

During an integration and acceptance process, the impact of the security of the system is significantly influenced by the overwhelming set of operational customer requirements. The integrator team and the customer team are frequently placed in an adversarial position due to cost and performance constraints. A significant number of large integrations are additionally stressed by being fixed-price contracts with little room for optimizing or trade-offs. As a result, the focus of the normal integration effort is on meeting immediate operational needs. The integrator's program management is frequently unwilling to make further trade-offs that unfavorably affect the cost of the effort or that change the end-user perception of the system's operability.

The integration contractor does not have the authority to trade between different customer policy and requirement sets. This task falls to the customer certification and accreditation team, which should be composed of operational as well as security personnel. The certification team executing the certification and accreditation plan should be composed early in the project; it should include a representative of the Designated Approving Authority (DAA), whom we shall call the accreditor. Having the certification team participate in developing the security ConOps and system security architecture, or alternatively developing the set of security requirements to be tasked in the SOW for the intended operational environment, provides the accreditor with insight into the rationale for the security approach. It also allows trade-offs to be made by the customer early in the development cycle. The accreditor's agreement should always be documented.

A logical extension of the TCSEC design and evaluation paradigm as given by Tinto [TINT92] is that the product composition paradigm can be viewed as being initiated by decomposing the system policy, then by allocating the system policy and functional responsibilities across the selected system products.

## Integration phase

**MLS integration policy needed.** There is no guidance for assessing the level of security in an integrated trusted system solution. Likewise, there is no nomenclature for even describing the level of technical security in such an integrated system. The efforts closest to a description are the product level descriptions (C1, C2, B1, B2, B3, A1) contained in the TCSEC and the modes (dedicated, system high, partitioned, compartmented, multilevel) of environmental description provided in the various DoD system policy documents. Security engineers need to know how to assess the level of trust of a system built from components. Lacking such a yardstick, the accreditor relies on instinct and experience in assessing the security of an integrated trusted system.

Much of this assessment is developed from the credibility of either the integrator's security organization or the customer's organization. Following Tinto again, the integration and assessment issue is to adapt the selected products and evaluate the adaptation of the system products in accordance with the specifications for each product in the context of the system policy.

Even when trusted components are used, the level of trust provided by the integrated system is unclear. The attributes of each component are affected by operation with other components. The unintended effects of their interaction are little understood. Trade-offs necessary to meet the customer's mission needs, and the revisions made in the respective components to achieve that result, further complicate the ability to accurately perceive the totality of security provided in a system. The various product security policies are, in many cases, different, resulting in even further difficulties. For example, products selected from different vendors by the integrator may have different approaches to labeling, auditing, and access control. Design decisions made by the independent vendor design teams are unconstrained by interoperability standards, resulting in invariable incompatibilities in format and policy enforcement. The result of the integration of these components may not provide adequate security controls for the system. Customer concepts requesting centralized authentication, audit, administration, and single-point (unitary) login each result in modifications to the various components that invalidate their individual evaluations. The impact of the adaptations or modifications on the integrated system solution is not defined and therefore cannot be reasonably assessed.

**Combine products securely.** Engineering a trusted system that meets requirements for functionality and trustworthiness is not simple or well defined. It involves identifying TCB alternatives such as:

1. whether and how to modify the TCB,
2. which trusted and untrusted processes can be modified and used,
3. how to convert existing applications, and
4. when to develop new applications [GAMB88].

In a custom solution to a secure system requirement, the first step is to decompose system requirements to the most detailed level possible, allocating software and hardware requirements into modules, components, or "boxes." The second step is to partition trusted and untrusted functionality for each requirement by determining which modules perform a security function in enforcement of security policy. Next come supplementary steps for trusted software, which include:

1. minimization of trusted code by iteration until minimum trusted code remains,
2. elimination of duplicate security functions,
3. verification and validation of trusted software, and
4. if new hardware is involved, code optimization for the new hardware platform.

The final step is then to recompose the policy to determine whether the intended policy is enforced by the combination of system products and adaptations.

For a COTS integration, the steps are somewhat different. The first step is to identify the functional “boxes” required in the operational system in a manner similar to a custom solution. The boxes likewise should not be constrained by the security requirement at this stage but reflect the best detail of the overall system requirements. Once the functional allocation is achieved, the information requirements for integration of those functional boxes need to be determined. These requirements may be expressed in the form of input, process, and output requirements, or in the form of a module interface definition.

Other approaches supported by formal security modeling may also be applied, such as information flow analysis. When these models are used, the policy enforced on that information flow between boxes can then also be described as either interface descriptions or input/output definitions. This description of information needs and information constraint should include elements essential to the security policy, such as security levels, access needs of users, and transmission across various media. Once this “architecture” has been validated, the security requirements should be overlaid to determine the functionality and level of product assurance needed to achieve that information transfer.

Integrating or composing two or more system components requires that an event in one component in some way causes a flow of information to an event in the target component. The method chosen to associate, or couple, events in the systems is the heart of TCB composition or trusted system integration. Possible levels of coupling range from completely disjoint to a completely coupled system in which all component policy elements are shared [FELL91]. Neither of these extremes is of practical use to the integrator. The real integration activity occurs in partially coupled cases. Partial coupling requires the precise definition of functionality and allocation of that functionality to the component.

Once the security functionality is established, the performance requirements need to be introduced into the solution. This late performance introduction ensures that the full set of operational requirements and security requirements is considered when calculating the hardware needs of the system. Invariably, the result will be a null set. This simply means that there is not available, as an evaluated solution, the precise

combination of applications, trusted products, and hardware that meets the requirements defined by the customer. This fact results in a set of trade-offs that must be considered by the integrator to meet the customer needs.

**Build or buy a TCB?** The TCB integrator's basic alternative is whether to build or buy. You should, as an integrator, buy whenever you can, as TCB development is very expensive, time consuming, and specialized. Basing the solution on a combination of NSA-evaluated TCB products listed in the evaluated products list (EPL) greatly simplifies certification, even though modification or addition will be required. A previously evaluated TCB can be modified, even though the modification destroys the EPL rating. Even with the EPL evaluation invalidated by adaptations, incorporating evaluated products into systems usually makes certification easier when the integrator is constrained by the contract in the formulation of the revisions. Building a new TCB is an integrator's last resort because of technical risk, expense, elapsed time, and certification difficulties.

The need to modify the product TCB occurs as a result of the trade-offs noted in the previous section in response to allocation decisions. In particular, the decisions regarding allocation of trusted features result in the need to modify the component TCB. The system-level security policy is used to ensure completeness of the functional security allocation. Once allocated to a component, the security function can then be compared with the component feature set. (This can be a recursive or reiterative process. The spiral approach has often been used to describe the allocation evolution.) Frequently, the component TCB contains features that are determined to be allocated to another component and not desired in this component. Additionally, the system may require a component to provide a function somewhat different from the product COTS capability. Note that if source code modification is necessary, it requires cooperation of the component TCB vendor. There are many business factors that might inhibit such cooperation.

You may elect to modify the component TCB by excising a feature of the component when that feature is to be provided in a different component. This is probably the more typical case of modification. You may amend the vendor software that provides a feature, when the system specification requires such a modification (for example, changing the password mechanism).

The third approach to modification may be to revise a vendor policy assumption used in the product evaluation (for example, identity based on TCB identity rather than user identity). This modification requires no revision to the TCB code but just as clearly invalidates the product evaluation.

**Trusted and untrusted processes.** Trusted processes are, by definition, part of the TCB. They are inside the TCB boundary because they enforce some customer security policy, or because they are exempt, in whole or in part, from mediation by other parts of the TCB. This exemption is necessary for the system to perform some element of the customer's intended mission. An "off-the-shelf" product's security policy often prevents necessary customer actions; this is circumvented by including selected "additional feature" modules within the TCB.

The concept of how a trusted process impacts on system integration requires some explanation. The common thought is that (a) because the processes violate the TCB policy, (b) they are trusted. However, these processes are not trusted because they violate the policy; rather they must be analyzed thoroughly to achieve a sufficient level of trust to be permitted to become part of the TCB. This means that the same level of assurance used for the product, or demanded by the system policy, must be applied to the design and development, evaluation, and configuration management of the "trusted processes" as defined for use in the system architecture.

Trusted processes are normally developed by the integrator as an adjunct to the COTS TCB. Trusted software must be analyzed extremely carefully for any undesirable interaction with other parts of the TCB. It is, therefore, highly desirable to minimize the amount of unique applications software with security relevance requiring insertion into the TCB. The process described in the section "Combine products securely" should be used to absolutely minimize the application requirement to that which is truly security relevant. Design review, redesign, and reanalysis should be iterated until you are satisfied that:

1. the amount of TCB code has been minimized,
2. the solution is necessary and sufficient to achieve only that minimum set of features required, and
3. the implementation is direct, concise, and easily assessable.

The untrusted processes are, by definition, outside of the TCB. These applications may have to be restructured to be integrated into a trusted system and continue to function correctly. This process is commonly called "porting the application," although normal portation is much simpler. You start by identifying all routines and software modules without security functions and continue by assessing compliance with system design objectives for all untrusted software.

Converting an existing application starts with selecting a TCB upon which to rehost it. Starting with a mature TCB is necessary to reduce the certification effort to permit the application to be installed as a single-level application in an overall trusted multilevel operation. After analyzing requirements to convert the existing software application to operate

under the control of the selected TCB, you discard the code that performs features which will be performed by the underlying TCB. As a result of this process, it will normally be necessary to make modifications to the application because of partitioning between trusted and untrusted software. Certain functions will move from untrusted to trusted software.

Some existing applications may not work because of security violations caused by design or performance shortcuts, such as direct memory or screen access. These violations can be eliminated by rewriting, if you have the application source code. Other security violations, such as attempting to open files at multiple security levels at the same time, may require so much redesign as to be prohibitively difficult. This situation is especially critical if the integrator is not familiar with the detailed design of the application and its internal documentation is inadequate.

**Porting untrusted applications to a TCB.** Porting an application to a TCB involves understanding the needs of the application and the services offered by the target TCB. In this discussion, we concentrate on meeting security needs. Differences in operating systems are extremely important and may pose considerable difficulty, but these differences are well-known porting problems outside the scope of this discussion.

All TCBs will include a DAC mechanism that provides at least permission bits for read, write, and execute capabilities for owner, group, and others, and may also include an ACL (access control list) capability. This capability allows access control to the level of specific users or, at a more general level, multiple groups of users. For instance, an ACL can be composed that allows a specific user read and write access to a file or the members of several groups read access to a file. Since most operating systems provide some form of DAC, there should be little or no difficulty in adapting the application to the DAC on the target TCB. However, the interaction between permission bits and ACL may be complex and difficult to administer securely.

The creation and maintenance of DAC group membership vary among operating systems. Some permit any user to create a group; adding or removing members is under the control of the creator (the owner) of the group. Other operating systems centralize group administration under a role such as information system security officer. Such differences may affect the installation and operation of untrusted applications.

Mandatory access control (MAC) is one of the significant security features on the TCBs in the B and A classes. MAC enforces a security policy based on labels. MAC-based systems permit multilevel operation, but the MAC policy rules may not be compatible with some applications. Applications that internally manage temporary resources or cross-reference documents may fail when constrained by the MAC policy. Invariably, enabling an application to understand security labels necessitates plac-

ing some subset of that application inside the TCB boundary. Precluding the application from being either in the TCB or obstructed by the MAC policy generally requires conceptual reorganization in the application.

During the identification and authentication login dialogue, the user establishes an initial current sensitivity level at or below his or her clearance. Most operating systems allow a user to change the current sensitivity level without requiring the user to log off. However, there is a wide variation in restrictions applied to the initial login level, such as the initial level being constrained to a minimum or maximum level. Such variation between COTS TCBs may only be an inconvenience or may be a major problem, depending on the amount of flexibility the customer has allowed the integrator.

A multilevel directory capability is provided in most trusted operating systems, but some do not provide an interface that allows an application to use this capability. In certain situations, the multilevel directory capability is an effective means of buffering the application from the effects of MAC, thereby easing the installation of an untrusted application. However, there are quirks in the implementation of multilevel directories on TCBs that must be dealt with.

Some applications record events in an application audit file. Trusted systems enforce access policies to the audit file that may not be compatible with the untrusted application. Further, MAC can require an audit log at each sensitivity level if a “write equal” policy is enforced. When such a restriction is levied, unanticipated by the untrusted application, unexpected application errors will occur. User dictionaries may “disappear” as a result of being modified by a higher level application domain. In addition, logical views of separate database tables may malfunction due to similar modification difficulties.

The effects of multilevel security are generally transparent in a single-level mode of application operation, but can cause problems when the application attempts to dynamically create files and write to files, due to the “no write down” aspect of MAC policy. For example, temporary work files can be a problem due to a mismatch between the user’s operating level and the directory in which the temporary file is written.

**Complex systems.** While developing trusted products is difficult, integrating evaluated trusted products (and some unevaluated products) into a real system to function in a real environment is even more challenging. If the system is sufficiently simple to permit implementation of a single security policy under a single security administration, the products possibly can be modified to meet the requirements for what the TNI calls a single trusted system. DODD 5200.28 uses the term unified system to refer to the same concept. In this concept, the integrator normally must demonstrate a single point of policy enforcement within the TCB which unambiguously performs each of the minimum functions of the

TCSEC or TNI. This simplistic integration then could be evaluated under the concept of a network product. However, the network product rarely fully meets the needs of a secure system's integration.

The administration of a large integration necessarily requires multiple administration points. The geographical dispersion of a large integration requires multiple contingency recovery approaches.

The network-based alternative is to approach the integration as an interconnection of separately accredited AIS. In this approach, each of the partitioned components would be assessed on its own merits. In a typical system integration, there is also little likelihood that the system would meet the definition of the TNI for this product evaluation. However, each of these separately accredited AISs must fully support accreditation on its individual merits. The allocation of partitions of policy requirements for the system TCB to individual components removes features from the components necessary for the independent accreditation of those components. Even in the case of interconnection of separately accredited AISs, the collection of components within the integrated system cooperates to collectively meet the intent of, or the set of features required by, the security accreditation policy. As stated earlier, there is no guidance for evaluation of an integrated system. The available guidance focuses on merely the evaluation of the underlying computer or network components.

The salient feature of systems integration is that the community needs additional guidance. Although two integrated systems may be defined as requiring the same level of trust (either TCSEC or TNI), applicable security features and assurances associated with these systems can vary significantly [FERR91].

## **Certification and accreditation**

In this section, we discuss the role of certification and accreditation (C&A) and establishing a C&A program.

**Role of certification and accreditation (C&A).** Certification is a formal extension of independent verification and validation (IV&V). That is, the organized process of quality control is separated from the process of system development and provided by independent persons. Certification provides a technical assessment of the security properties of a system relative to criteria selected at system inception. This process is actually the recomposition of the "system policy to determine that the intended policy is enforced by the combination of system elements" [TINT92]. The certification process then requires evaluation of the integrated system operating in its environment. The purpose of this process is to ensure

that all of the criteria have been satisfied at some standard level. The technical certification report is a valuable input to accreditation.

Accreditation involves an assessment to determine if a system in its operational mode has minimized risks to a level that is secure enough to permit the system to process operational data. While certification is a technical assessment of the adequacy of the features of the system, accreditation is a management decision that balances operational mission criticality or need against the residual risk. This decision is made also considering:

1. the sensitivity of the data,
2. cost-effectiveness of additional enhancement of features or assurance,
3. definition and enforcement of administrative procedures, and
4. evaluation of the adequacy of the security administration and end-user training program.

The accreditor makes the case-by-case decision to permit the system to operate within a given set of operational constraints. These constraints are normally expressed as mode of operation. Only the dedicated mode processes a single level of information; system high, partitioned, and multi-level secure modes all simultaneously process multiple levels of information. Only the protection requirements or user/facility clearance environments are differently constrained.

The accreditor has other options, among which are:

1. reject the system for operational use,
2. modify the mode of operation to a mode having a lower risk,
3. redefine the operational environment by restricting potential users, or
4. accept the system for operational use, conditional upon progress being achieved on correcting the deficiency.

The ITSEC emphasizes that accepting a system for use in a particular environment requires consideration of multiple factors before the system can be considered as fit for its intended purpose. Considerations include:

- sensitivity of data processed,
- user trustworthiness,
- intended mode of operation (for example, dedicated, system high, multilevel),
- assurance in the technical security provided by the system,
- the owner of the information,
- confirmation of management responsibilities for security,

- compliance with relevant technical and legal/regulatory responsibilities for security,
- confidence in the adequacy of nontechnical security measures provided in the system environment, and
- the mission and operational concept.

The accreditor has the following options:

- permit operation as planned;
- require specified changes in system design or implementation, the way the system is operated, or the environment before operation is permitted;
- permit operation for a limited period on the condition that specified changes are made within that period;
- restrict the mode of operation (for example, deny multilevel operation but permit partitioned mode); or
- prohibit operation.

The available official guidance [GUID83, GUID84] is not current; some papers are available (see [BAUE91] and its references) that report on actual experience and lessons learned.

## **Summary**

We have shown that the transition from product to workplace requires a set of ideals different from that of building a trusted or evaluated product. The dominance of security in the development of a product is replaced by the functional and performance requirements of a system statement of work or contract specification. The challenge is that of working in an environment which has no agreed on guidance or standards. This situation can be both frustrating and risky. The keys to achieving a successful system integration are the cooperation of the accreditor in establishing the requirements before contract solicitation, the skill of the integration teams in developing approaches that minimize risk, and the skill of the customer in choosing the winning approach.