The Design and Evaluation of INFOSEC Systems:

The Computer Security Contribution to the Composition Discussion

Mario Tinto

Library No. S239,214
June 1992

FOREWORD

This C Technical Report, "The Design and Evaluation of INFOSEC Systems: The Computer Security Contribution to the Composition Discussion," is issued by the National Computer Security Center (NCSC) under the authority of and in accordance with Department of Defense (DoD) Directive 5215.1, "Computer Security Evaluation Center." This publication contains technical observations, opinions, and evidence prepared for individuals involved with computer security.

Recommendations for revision to this publication are encouraged and will be reviewed periodically by the NCSC. Address all proposals for revision through appropriate channels to:

National Computer Security Center
9800 Savage Road
Fort George G. Meade, MD 20755-6000
Attention: Chief, Standards, Criteria & Guidelines Division

Reviewed by: __________________

RON S. ROSS, LTC (USA)
Chief, Standards, Criteria & Guidelines Division

Released by: _______________________

THOMAS R. MALARKEY
Chief, Office of Information Systems Security
Standards, Assessments, & Systems Services

Table of Contents

I. INTRODUCTION

There has recently been a heightened awareness within Information Security (INFOSEC) organizations of the absolute need to approach systems security issues from a more global perspective than had previously been the case. A major consequence has been the investigation of "the composition problem"; that is, how to bring a set of components together to form a system and then be able to determine the security attributes of the system from the analysis of the properties of the system's components. And rightfully so. This is, arguably, the central issue that must be dealt with by any organization claiming to provide INFOSEC services. Anyone who has ever had to provide security support to systems of even moderate size and complexity understands the scope of the problem. One is typically faced, on the one hand, with a set of high-level, sometimes abstract, requirements for the system. On the other hand, the system will often be a combination of commercial products, government-provided products, and special-purpose devices and/or software developed by the system integrator. It is not unusual for the program office to articulate state-of-the-art functional and security requirements while being constrained to products already in their capital inventory, making the satisfaction of many of the requirements a challenging, if not daunting, task. In any case, there is a substantial intellectual gap between the high-level statement of requirements and the realities of the implementation. Thus, the essential question is, "what global properties can be asserted about a set of heterogeneous hardware and software products that have been brought together to provide services and resources to a community of users?"

Recent developments in the computer security community have dealt with these issues from a slightly different, but relevant, perspective. The insights gained appear in the "Interpretations" of the Trusted Computer Systems Evaluation Criteria (TCSEC); the Network Interpretation (TNI), and the DataBase Interpretation (TDI). This paper will examine the contributions made by this set of work, and then discuss the extension of the insights provided to the more general problem of INFOSEC systems.

II. BACKGROUND: The Computer Security Fundamentals

In order to proceed, it is important that the set of computer security documents (i.e., the TCSEC [10], the TNI [11], and the TDI [12]) be seen and understood as a complete body of work. Whereas the TCSEC introduces and discusses fundamental issues, the other documents provide insights concerning the application of basic principles to complex systems. Thus, the TNI and TDI are not, in fact, separate documents. Rather, they complement and extend the TCSEC.

This section sets the framework for the later discussion by introducing the foundational material: the security issues to which the documents are directed, the technology basis for addressing those issues, and the fundamental concepts and premises that underlie the documents.

II.a The Sharing Issue

The first question that must be asked is, "what problem or set of problems is being addressed?" The issue with which the TCSEC deals is illustrated by figure 2.1, which depicts two computer systems, each being shared by a number of users. In the first system, all users have the same authorizations for the information and services supplied by the computer system. This is indicated by showing all (both) users being cleared at the SECRET level. Thus, there is no compromise issue involved, and all the protection required can be obtained via physical and procedural approaches?guards, guns, & gates. That is to say, that as long as physical access to the computer, communications, and terminals can be limited to only authorized (in this case, SECRET-cleared) persons, sufficient and adequate security has been achieved. However, in the second example, we have a system for which not all the users have the same authorizations for resources and information. This is shown by indicating that one user is cleared to the SECRET level (implying, of course, that there is information at the SECRET level being processed by the system), whereas the other user is not cleared. For this case, no amount of external controls prevent the uncleared user from obtaining information for which he is not authorized. That problem can only be addressed by including mechanisms which are internal to the computer system itself. Note that the use of classifications is only illustrative. The problem is the same if we postulate two persons (or organizations) who consider their material sensitive and to be protected from arbitrary access by others who are also using the system.

The Issue is Sharing

Thus, the technical issue that must be addressed is that of sharing. In other words, given a system that provides computational capability?services and resources?that are shared among people of varying authorizations, how is access to those services and resources controlled such that users receive no more than that for which they are authorized? This is fundamentally a computational problem, to be dealt with via computer science.

II.b The Reference Monitor Concept

Now that there is clear statement of the problem, we can proceed to ask what can be done about it. The early years of the computer security experience can be characterized by a "penetrate and patch" approach?systems were tested to uncover flaws, and the penetration paths uncovered were then patched. The system was then re-tested, invariably revealing still more (or possibly new) flaws. This is an unsatisfying process for developing secure systems, largely because it is not clear when such a process is completed. Is it completed only when no more flaws are uncovered? Does the absence of flaws say something about the strength or quality of the system, or about the skills and endurance of the tester?

What is really wanted is a general solution. That general solution is commonly known as the "reference monitor concept," and was first articulated as the result of a 1972 Air Force study [1]. The Air Force study in essence asked the question, "what kind of computer system can be built that guarantees that security is preserved regardless of the (possibly hostile) actions of users?" The resulting report (i.e., "the Anderson Report") described an architectural framework for dealing with mediation of access in the face of potentially hostile users.

The essence of the reference monitor notion is depicted in figure 2.2. In this model, the entities of interest in a system are "subjects" and "objects." A subject is an active entity, loosely described as a program in execution, and is the surrogate of a person. Thus, a subject has an identity and attributes. An object is a passive entity, usually a repository of information. The reference monitor is an abstract machine that reliably mediates the access of subjects to objects. On each attempted access of an object by a subject, the reference monitor determines whether or not the access is to be granted. It does this by applying a set of access control rules along with information it has about the subjects and the objects.

The realization of such an architectural construct in hardware and software is what the TCSEC refers to as a "Trusted Computing Base (TCB)." Note that the reference monitor is a conceptual model; we have yet to discuss implementation considerations. Clearly it must be the case that a real machine must have demonstrable properties which provide the requisite level of confidence that it is doing what was intended. Accordingly, an implementation of a reference monitor must have the properties that:

(a) It is always invoked; it cannot be bypassed. All accesses are mediated.

(b) It is tamper-proof; nothing a subject (user) attempts can either interfere with the reference monitor or alter its internal code, data structures, or the databases on which it relies.

(c) It is small enough to be subject to analysis and tests. In other words, it must be conceptually simple.

These properties must be demonstrated, and they are usually argued on the basis of the hardware and software architectures.

The General Solution to the Sharing Problem is the Reference Monitor Concept

In summary then, the TCSEC deals with the problem of sharing; how to mediate access to data and resources by and among a group of individuals with varying authorization and rights. And, the relevant architectural approach is that of the reference monitor?the assured and unassailable enforcement of a specified set of access control rules.

II.c The Design & Evaluation Paradigm

One final notion that is central to the TCSEC remains to be covered, that of the relation between policy and mechanisms. In the TCSEC parlance, the term "security policy" is defined as, "the set of rules, directives, and practices that regulate how an organization manages, protects, and distributes sensitive information." A security policy is then translated into a set of access control rules that are to be enforced by the TCB. The desired attributes of the system are eventually realized, in part, by the implementation of some specific set of "mechanisms," functions which can be shown to provide the requisite attributes. The critical point is that, as shown in figure 2.3, one proceeds from policy (i.e., a high-level statement of the desired global properties or characteristics of the system) to a set of specific mechanisms. It is only when one knows what global characteristics the system must have that it becomes possible to evaluate the implementation details. It is only in the context of the statement of policy and the subsequent access control rules that the chosen mechanisms can be evaluated for both sufficiency (are there enough to do the job?) and adequacy (do they provide the effects desired?). This may appear intuitively obvious, but it is critical. Systems that state "requirements" in terms of the set of desired features end up making circular arguments; one argues from mechanisms to mechanisms. This approach leads to tautologies (i.e., "the mechanisms provided are the mechanisms that were specified") which reveal nothing about the properties of the system that are truly of concern.

Design and Analysis of Systems Proceeds from Policy to Mechanisms

Now that we have established the fundamental concepts and principles, we can proceed to examine how these few simple ideas are applied to the design and analysis of systems.

III. ARCHITECTURAL CONSIDERATIONS: The TNI and the TDI

Historically, TCSEC evaluations have been approached with the view that a TCB is an atomic entity; it is monolithic and homogeneous. This view has major consequences, effecting how evaluations are carried out and the requirements levied upon those who request evaluation of their systems. A discussion of the details of the technical and business implications of such a view are beyond the scope of this paper. However, what is important is the recognition that most systems of interest tend to be relatively complex. In particular, we are concerned with systems that exhibit some structure; they are made up of a collection of sub-elements. The problem to be addressed then, is whether or not the collection of sub-elements, working as a system, provide the global attributes and properties that are desired?do they compose?

The problem presented by large, complex systems, however, is one that may be characterized as "scope of comprehensibility"?one is faced with comprehending the details of a number (often a large number) of functional elements, but what has to be known are the global system properties that they exhibit when they all work together. It is difficult to get one's mind around the entire system at once, but the only path to understanding the system is to deal with the elements of the system. Unfortunately, the elements of the system present the evaluator with details that are related to, but may divert attention from, the system properties that are to be determined. Thus, the problem of assessing system properties is compounded by an "intellectual gap"?the level of abstraction presented by the elements (i.e., implementation specifics, functional characteristics) is different from the level of abstraction that is of interest (e.g., security). The problem then, for both designers and evaluators, is how to approach a complex system such that the elements derive from the high-level system objectives (in the TCSEC context, enforce the security policy) and that they can be shown to compose to those objectives.

III.a Partitioned Systems

The first class of structure that will be dealt with is partitioning. A partitioned system is characterized as one that is composed of cooperating, peer-entity elements, as shown in figure 3.1. That is, the responsibility for enforcing the system policy (or, more generally, for satisfying the system requirements) and for providing the desired functions is allocated across the system elements; they "cooperate" to provide the capabilities desired. Additionally, no element has more privilege or status than any other. This architecture is addressed in the Trusted Network Interpretation (TNI, commonly referred to as "the Red Book"). The purpose of the TNI is to demonstrate that the TCSEC principles are extensible. Specifically, to demonstrate that they extend to partitioned systems in general and networks in particular. A common misconception is that the TCSEC applies strictly to operating systems of "stand-alone" computers. However, a communications network also represents a shared computational system. The resources and services a network provides may be different from those presented by general-purpose computers, but it is providing computational services and performing access control on shared resources nonetheless. A general-purpose computer provides and mediates access to entities such as files and devices and provides services which allow manipulation of information and data, whereas a network will present its users with communications services and access to entities such as messages and ports. Thus, the principles and methods of the TCSEC are sufficiently general to apply to useful and interesting aspects of network security.

Because the notions "networks," "partitioned systems," and "distributed systems" are often confused and used interchangeably, it is important to recognize that clear distinctions can be made between them. These concepts are neither synonymous nor mutually exclusive. A partitioned system is one in which the various functions to be performed by the system are realized as a set of separate (but interrelated) elements, or modules that communicate via mechanisms such as parameter-passing or message-passing. What is critical to the notion of partitioning is that each element, or module, represents a different function; that is, the system is not a set of replicated (i.e., identical) elements. The clear consequence is that no subset of the elements constitutes the system described by the requirements. Only the entire collection of elements completely satisfies the system specifications. Note that there is nothing in the definition of partitioning that requires physical separation of the elements. Additionally, as noted in the earlier definition, there is no hierarchical ordering of the elements.

It is not the purpose of this paper to enter into an extended discussion of each of these terms?only to clearly characterize the notion of partitioning. Suffice it to say that it is possible for any product or system to exhibit a partitioned architecture.

The TNI view for dealing with partitioned systems is depicted in figure 3.2. It is an extension of the top-down (i.e., policy to mechanism) approach discussed in section II.c, with a view to the problem of composition. Starting from the perspective of the system policy, policy and functional responsibility are allocated across the set of system elements; the policy is decomposed. This process yields a specification for each element, each specification being derived from the system requirements. Now, each of the elements can be developed and evaluated separately ?against the set of requirements specific to the element in question. Then, after the system elements have been developed and evaluated for compliance with the responsibilities allocated to it, one can determine whether or not the set of elements provide the characteristics desired for the system; do the elements recompose to provide the policy defined for the system?

Let us examine a more concrete example, one of a TCSEC product. Presume that the set of access control policies (i.e., the discretionary and mandatory policies) is allocated to two separate elements, that the identification and authentication (I&A) function is allocated to still another element, and that the auditing responsibilities reside in several elements (e.g., an "event filter," a function that writes records to the audit trail, and a set of tools for post-processing of the audit data). There will be other functions, but this set will suffice for the example. Note that these functions are not independent; there will be functional dependencies between elements. Certainly, access control cannot occur without knowing the identity of the user who is requesting services or information. Thus, there will be a functional interface between the policy-enforcing elements and the I&A element; these two functions must communicate. Likewise, there will be interfaces between the various pieces of the auditing function, as well as an interface between I&A and auditing (because the user's identity needs to be part of the audit record). In short, the analysis of a partitioned system involves an assessment of the sufficiency and adequacy of the set of elements, the correct functioning of each of the elements, and the correct communication between interdependent elements. As we will see later, the analysis of hierarchically-ordered systems is conceptually simpler because the relation between system layers can be characterized solely in terms of access control policy.

The essential aspect to this approach is what amounts to a re-definition of the composition problem. The notion here is decomposition and eventual recomposition of the system policy. Note that the reason one can argue that the elements compose to the system policy is precisely because the responsibilities for each of the elements were originally derived from the desired system properties; the specifications for each of the elements are established in the context of the defined global properties of the system. The claim here is that, in general, one cannot ask about the composition of an arbitrary set of elements?one does not interconnect a VCR, a carburetor, and a toaster, independent of any system context, and then ask, "what policy is enforced?" Composition does not occur by accident!

In summary, the way one proceeds is:

(a) decompose the policy; allocate policy and functional responsibilities across the system elements;

(b) develop and evaluate the system elements in accordance with the specifications for each; and

(c) recompose the system policy; determine that the intended policy is enforced by the combination of system elements.

III.b Hierarchical Systems

The next structure to be considered is that of hierarchical ordering. A hierarchically-ordered architecture is characterized by dependency. A simple case of a system exhibiting hierarchical dependencies is illustrated in figure 3.3. For such a system the notion of "primitivity" is useful. An abstract machine B is "less primitive" than an abstract machine A if (1) B directly depends on A, or (2) a chain of machines exist such that each machine in the chain directly depends on its successor in the chain. Thus, in figure 3.3, machine 2 is "more primitive" than machine 3 because machine 3 obtains resources and services from machine 2, and is dependent on machine 2 for its own correct functioning (that is, if machine 2 does not function correctly, than it cannot be argued that machine 3 functions correctly). Machine 1 is the most primitive, because it neither depends on any other machine for correctness, nor does it obtain resources from another machine; it "owns" its own resources. This is the class of architectures that is dealt with in the TDI.

Unlike the case for partitioned systems, there is a clear "privilege" hierarchy. This, as we will see later, allows for strong conclusions to be reached under certain conditions.

The TDI was largely motivated by fairly pragmatic considerations, but it is the implications on system design and analysis that are of consequence. The TDI was driven by the need to be able to evaluate the kinds of products that were expected to appear in the marketplace. In particular, some applications, such as database management systems (DBMS), transaction processors, and electronic mail, also have the need to control the sharing of resources and services?they need to provide security capabilities. Insofar as a system deals with access control issues, the principles and technology of the TCSEC are relevant. However, it is not unusual for the vendor of an application, such as a DBMS, to provide a software product that is intended to be hosted on an existing platform consisting of hardware and probably an operating system, as illustrated in figure 3.4. A practical problem arises when a vendor submits for evaluation a product to be hosted on an existing platform that has already been evaluated. In effect, the vendor is proposing for evaluation a system composed of distinguishable parts, one that is known, and one that is new and unknown.

The procedures of the TCSEC evaluation process were predicated on the premise that evaluations would be focused largely on operating systems. That is, the system to be evaluated was expected to be entirely the product of a single manufacturer. Thus, the process that existed when the TDI was being developed required the vendor to provide all the evidence required to demonstrate that his product satisfied the requirements of the target rating. This would include, among other things, information about the hardware. This is certainly not unreasonable; the evaluator must be able to ascertain that the product is indeed what the vendor claims it to be. However, this approach has the consequence of requiring an applications vendor to provide proprietary information about the host system; information which he may not be able to obtain, or may be legally restricted from releasing. It would be advantageous to have a method whereby the vendor need only present information about those parts of the composite system that are new. Also, from the point of view of the evaluator, we are faced with the prospect of evaluating the entire system, major elements of which may already have been evaluated and whose properties are already known. Thus, it would be advantageous to be able to reach accurate conclusions about the composite system by evaluating only the new elements; that is, without replicating previous evaluation efforts. This is exactly the composition problem! In other words, the question is, "under what set of conditions is it possible to determine the global characteristics of a system as a result of the separate assessment of distinguishable parts of the system?"

In short, the TDI was focused on the set of problems: extension of TCSEC principals to applications, conservation of evidence, conservation of evaluation effort, and the composability of evaluations in a layered (or hierarchically-ordered) system.

TCB Subsets

The primary logical construct for dealing with the problems presented by layered (or hierarchically-ordered) systems is that of "TCB subsets" [15]. The notion of TCB subsets is a conservative extension of the reference monitor concept. That is, the original definition of a reference monitor implementation (or a TCB) included the hardware, whereas a TCB subset is defined such that it may, but need not, include hardware. However, a TCB subset has the essential characteristics outlined in Section II.b. That is, it enforces a defined policy?it mediates the access of a set of subjects to a set of objects on the basis of stated access control rules. Additionally, it must be shown to be tamper-resistant, always invoked (i.e., mediates every access), and small enough to be subject to test and analysis, the completeness of which can be assured.

The important consequence is that there is a reference monitor per layer of the system; each layer (or, subset) is complete with respect to a policy enforced over a set of resources exported by that layer. This, in turn, means that layer m is constrained by the policy enforced by layer m-1; layer m is untrusted relative to layer m-1. It is precisely this condition (i.e., that each layer is complete with respect to policy) that allows one to advance a convincing argument about the composability of the set of system layers. At the lowest level (TCB Subset0 in figure 3.5) there is a set of hardware, software, and possibly firmware that implements the reference monitor concept (i.e., it constitutes a TCB). By definition, nothing that happens external to this layer may violate or bypass the security policy enforced by it. Clearly, this assertion is the focus of considerable analysis. However, the critical point is that the analysis of this layer may be undertaken independent from, and without regard to, the analysis of any other element of the system; no action of any higher (or less primitive) layer can alter the results of the analysis of this layer. The next layer of the system (i.e., TCB Subset1) treats the previous layer as merely an abstract machine (or "virtual hardware"), providing resources and exhibiting hardware-like properties (e.g., support for distinct domains of execution). It creates the abstractions of subjects and objects over which some, possibly different, access control policy is enforced. As before, the analysis of this TCB subset can proceed independently; whether or not this level of the system satisfies its requirements is knowable only from the details of this level and the properties exported to it by the abstract machine (i.e., the more primitive layers) upon which it depends. The concept is recursive; each layer treats the more primitive layers as an abstract machine from which it obtains resources, and by whose policy it is constrained. In turn, each layer exports services and resources which are mediated in accordance with the set of access control rules it imposes. A more detailed discussion is presented in Appendix B, "Reasoning About TCB Subsets."

Thus, the way in which one proceeds for system design and analysis is:

(a) decompose the policy; allocate policy responsibilities across the system layers;

(b) develop and evaluate the system elements in accordance with the specifications for each; and

(c) recompose the system policy; determine that the intended policy is enforced by the combination of system elements.

Note that this is consistent with the procedure for dealing with a partitioned system, as discussed in paragraph III.a. That is, one starts from the global requirements for the system, derives requirements for each of the elements of the system (i.e., each of the layers), analyzes each of the elements for compliance with its derived requirements, and ultimately decides on the reasonableness of the assertion that the properties of the elements compose to the properties desired of the system. However, the one significant difference is that resulting from the condition that each TCB subset is complete with respect to policy. This constraint means that the interaction between subsets can be completely characterized in terms of policy; the functional dependencies that add complexity in the case of partitioned systems do not present the same problem with the TCB subset approach to hierarchical systems.

To summarize the TDI and the TCB subsets approach, we now have a method for dealing with hierarchically-ordered architectures?it provides a way to define the system as a set of relatively independent system layers which, in turn, provides the basis for arguing about the composition of the system.

IV. THE COMPLETE PICTURE: A SYSTEM ARCHITECTURE

We can now put the pieces together. The concepts of the TNI, dealing with partitioned systems, and those of the TDI, dealing with hierarchically-ordered systems, together provide an elegant and powerful set of notions for dealing with systems of arbitrary complexity, as illustrated in figure 4.1. At the lowest level of the system, an access control policy is enforced on a "primitive" set of system resources (e.g., segments, pages). At the next level, two different operating systems are implemented (e.g., UNIX, DOS), each of which export different abstractions and possibly different policies. At the highest level of the TCB, several utilities are implemented. For example, one can imagine an e-mail service on O/S 1 that implements access control on entities such as messages and mailboxes and a transaction processor on O/S 2 that controls access to executable code (or, "canned" transactions), as well as a database management system that controls access to entities such as tuples, rows, and columns. Each of the entities shown could, in turn, exhibit a partitioned structure (however, for the sake of simplicity, only the first layer (i.e., "low-level services") is shown as such). Although this is a fairly straightforward computer system example, one could just as easily postulate a command and control system in which the lowest level is a distributed network that controls the access of "hosts" to "messages." At the next level, host computers implement host-specific policies (e.g., Clark-Wilson [9, 10], Bell & LaPadula [3], Biba Integrity [8]) on entities such as files, directories, and devices. Finally, as above, each host computer provides a set of services, or applications, that may add further policy constraints or apply the existing constraints to different classes of objects.

V. EXTENSIBILITY TO INFOSEC SYSTEMS

In summary, we have theory and procedure for both the designer/developer and the evaluator. The designer can structure a system to provide the user interface(s) that satisfy both functional requirements and security policy requirements, and that structure has sufficient detail and granularity to proceed with the design and development of the system components. The evaluator, starting from a high-level statement of the system security requirements (i.e., the security policy), can proceed from an analysis of the component elements of the system to ascertain compliance with the desired global attributes of the system; he can argue about the composition of the pieces. The critical question, of course, has to do with the utility and limitations of these results to INFOSEC systems.

On the one hand, computer systems are, after all, systems. And, many of the INFOSEC systems we deal with are largely computer-based and provide computational services. The principles of the TCSEC and the results of the TNI and the TDI are quite general, applying to any computational system that supports (and must control) sharing. Additionally, we have seen that the principles, as well as the design and analytic techniques, are applicable to familiar and useful architectures. Most importantly, the techniques allow one to argue about composition.

On the other hand, the concepts presented herein were developed in the context of computer systems, access control policies, and the reference monitor concept. INFOSEC systems can be more complex, not so much in that their structures are more complex, but that they introduce security issues that are not easily representable as access control policies (in fact, they often are not access control policies). That is, INFOSEC systems introduce "policy complexity." For example, it is not clear that it is reasonable, or even useful, to discuss the general field of cryptography in access control and reference monitor terms. Likewise, topics such as "non-repudiation" and "fault-tolerance" introduce issues for which the reference monitor approach may be irrelevant. This is not to say that the decomposition and recomposition analysis and arguments which have been presented are not useful, but the composition argument becomes more complex and may be on less secure or, at least, less well-established theoretical foundations.

VI. CONCLUSIONS and OPPORTUNITIES

The principles and the design and analysis techniques represented by the TCSEC and its attendant "Interpretations" are important and useful results. They apply to a class of system problems and system architectures that are encountered daily. The challenge is to extend the results to INFOSEC systems in general. Minimally, it is incumbent upon us to determine the practical limitations of these results, and identify the research topics. In particular, the following areas of investigation are recommended:

(a) The case for the composition of system elements herein has been informal and has been made largely as an appeal to intuition. Some work has been done in the theoretical foundation for the composition of layered systems [13]. However, formal analysis of the composition of partitioned systems has yet to be pursued.

(b) As has already been noted, the manner in which requirements can be expressed may limit the scope of theoretical results. Some requirements may be expressible in terms that allow them to be easily and naturally incorporated into the existing theory and framework. For other requirements, it will be necessary to extend the theory.

APPENDIX A: TCB Subsets and Partial Ordering?

The presentation of section III.b was limited to systems that are strictly hierarchical. However, this was done merely for simplicity of presentation. In fact, the concept of TCB subsets is considerably more general, allowing one to deal with architectures that reflect partial ordering.

The definitions that are central to the concept of TCB subsets are those of "primitivity" and "dependency." These terms were introduced in section III.b, and will be expanded on only briefly here. A more lengthy and rigorous discussion is provided in Appendix B of the TDI. The key notion is that of "dependency," which can be loosely described as follows:

Consider two systems (or abstract machines) M1 and M2, with specifications sM1 and sM2 respectively. The usual question of interest is how well a machine, M, does what its specifications, sM, say it is supposed to do. That is, the degree of correspondence between sM and M. We say that M2 "depends on" M1 if the arguments about the correct implementation of M2 with respect to sM2 must assume the correct implementation of M1 with respect to sM1. That is, M2 "depends on" M1 if the correctness of M2 cannot be demonstrated without assuming (or demonstrating) the correctness of M1.

As an example, consider a set of communications protocols. Each layer is, in fact, an abstract machine with a corresponding specification. Clearly, arguing about the correctness of any layer depends upon the correct implementation of the previous layer(s). For instance, a layer responsible for packet reassembly must assume that some lower layer has made the correct correspondence of packets to "connection." Note that dependency is not the same as information flow. In the example of the protocol stack, data and parameters flow in both directions, but the dependency is "downward." Nor should dependency be confused with the accuracy of input data ? a calculator can be shown to correctly implement its specifications regardless of the values that are presented to it.

As a related note, modern software engineering tends to be concerned with "circular dependencies," which occur when machines are mutually dependent. These are cases to be diligently avoided, and to be corrected when they occur, precisely because they make it difficult, if not impossible, to reason about the correctness of the resulting system.

The question now is, "what are the architectures that can be formed using only the concepts of primitivity and dependency?" It turns out that it is the class of architectures that are partially ordered (i.e., exhibit a "tree-like" structure), as shown in figure 1. In the illustration, the arrows indicate the direction of dependency. Thus, M1 "depends on" M0 (designated by "M1 ? M0"). Note that both M1 and M2 depend on M0 (or alternatively, M0 is more primitive than both M1 and M2). This structure is consistent with the notions of primitivity and dependency and only means that both M1 and M2 obtain resources and data from M0, and that neither M1 nor M2 can be shown to be correct without also showing that M0 is a correct implementation of its specification. There is no implication about dependency between M1 and M2. In fact, in the illustration it is clear that there is none?they are independent; neither is dependent upon the correctness of the other for its own correct behavior (even though they may exchange information). The relations in the illustration are:

(a) M3 ? M1 ? M0

(b) M4 ? M2 ? M0

(c) M5 ? M2 ? M0

Note that these relations are transitive. Thus, from (a) it is also true that

M3 ? M0.

Alternatively, M0 is more primitive than M1, which is more primitive than M3. Consequently, from the definition of "primitivity" given in Section III.b, M0 is more primitive than M3.

Thus it is seen that the notion of dependency is sufficient for describing architectures that exhibit partial ordering (i.e., a set of machines or modules that are organized as the nodes of a rooted tree). As a consequence, the construct "TCB subsets" is also applicable to such structures.

APPENDIX B: Reasoning About TCB Subsets

This appendix is intended to provide a deeper understanding of the TCB subset concept by demonstrating how it is applied and how the central arguments proceed. The references to the levels of the TCB are keyed to figure 3.5.

The most primitive level is TCB Subset0, which is composed of hardware, software, and possibly firmware. The hardware exports entities such as words or bytes and typically provides for distinct execution domains through the hardware structure and memory management functions. To demonstrate the principles involved, it will be useful to postulate a hardware that provides four domains, or states (e.g., Intel i80286, i80386). That is, code may execute in any of four hardware states, with varying attendant "privileges" and capabilities. The software and firmware portions of this layer create a higher level abstraction, such as segments or pages that it exports (i.e., presents at its external interface). It also imposes a set of access control constraints (i.e., enforces a set of rules that determine the conditions under which the exported abstractions may be accessed and manipulated). Finally, this layer will export hardware or hardware-like properties; it may provide unused hardware domains for use by other layers, or it may provide logical domains. For this example we will assume that TCB Subset0 executes entirely in the most privileged hardware state. It makes the remainder of the hardware states available to untrusted (relative to itself) processes and, at the same time, constrains all processes not in Subset0 to the less privileged states. That is, no subject that is untrusted relative to TCB Subset0 may execute in the domain of TCB Subset0. This is precisely what allows a convincing argument to be made about the "tamper-proof" property of this TCB Subset. More to the point, it is necessary to demonstrate that this layer satisfies the definition of a TCB Subset?that it implements the reference monitor concept and has the essential properties of a TCB. The properties "always invoked" and "analyzable" are argued from the software architecture and the correct and complete implementation of the interface specification. The essence of the argument about these properties is the same at all layers. That is, these arguments proceed from the software and its implementation. Because the property of tamper-proofness is traditionally argued from the attributes of the hardware, introduction of the concept of TCB Subsets makes this the property that requires special attention. The most primitive layer (i.e., TCB Subset0) contains hardware, and thus the argument for tamper-proofness proceeds in the conventional manner, based upon the ability of this layer to control the hardware and thus guarantee that only its processes execute in the most privileged hardware state. In turn, this means that no processes external to this layer can modify the code or data structures of TCB Subset0. The ramification for evaluators is that once the analysis of this layer is complete, one does not have to return to it, because nothing that is implemented at less primitive (i.e., higher) layers can change the results obtained. For example, the penetration testing and covert channel analysis are carried out at the interface, or policy boundary, enforced by TCB Subset0. Because any higher layer can only obtain services and resources through this interface, no higher layers can (assuming, of course, that the analysis was complete) do anything that is not already accounted for in the analysis. In other words, the higher layers can be no more malicious than the evaluators have already been allowed to be. As a consequence, the analysis of this layer can be performed independently; analysis of the properties of this layer provides evidence about the composite system that cannot be affected by any less primitive layer.

The next layer (i.e., TCB Subset1) is implemented as a set of processes (or subjects) that are untrusted relative to the more primitive layer, TCB Subset0. This means that TCB Subset1 is presented with the objects exported by TCB Subset0, its access to them is constrained by the policy enforced by TCB Subset0, and the processes that make up TCB Subset1 are further constrained such that they may not execute in the most privileged hardware state. What this layer has available to it is the remaining three hardware states. Briefly, it "views" TCB Subset0 as though it were merely hardware, providing services and resources in accordance with an explicit interface definition. That is, TCB Subset0 is treated as merely an abstract machine with hardware-like properties. In the engineering vernacular, it is a "black box." As noted above, the property of interest is that of tamper-proofness. For TCB Subset1 it is argued from the fact that this layer has available to it a domain structure that is enforced by the "virtual hardware," TCB Subset0 (and ultimately by the actual hardware). Thus, TCB Subset1 may reserve for its own execution the most privileged of the states available to it, in turn constraining all subjects supported by it (i.e., all "untrusted" processes) to execute in the less-privileged domains. Thus, a convincing argument can be advanced that no processes that are untrusted relative to TCB Subset1 (i.e., processes of any less primitive layer) can modify the code or data structures of TCB Subset1. TCB Subset1 will use the abstractions (e.g., segments) presented to it to create other abstractions (e.g., files, directories) that it, in turn, makes available at its interface. Additionally, it will impose a set of access control rules on these objects. Thus, the arguments that TCB Subset1 does, in fact, represent a valid implementation of the reference monitor concept proceeds in a fashion that is a conservative extension of the traditional line of argument. And, as before, the analysis of this layer can proceed independently.

Clearly, the process is recursive. Each layer views the next more primitive layer(s) as a set of virtual hardware, exporting services, resources, and properties. The property of tamper-proofness can be argued if, and only if, it can be shown that the layer in question has available to it a domain for its own execution, from which it can control the action of the (untrusted) subjects it supports.

Finally, because each layer is tamper-proof relative to all less primitive layers, one can perform analysis on each of the layers separately and still reason to the composition of the system policy.

REFERENCES

1. J. P. Anderson, "Computer Security Technology Planning Study," ESD-TR-73-51 (AD-758206), J. P. Anderson Co., October 1972.

2. D. E. Bell and L. J. La Padula, "Secure Computer Systems: Unified Exposition and Multics Interpretation," MTR-2997, (AY/W 020 445), The MITRE Corporation, Bedford, Massachusetts, July 1975.

3. D. E. Bell and L. J. La Padula, "Secure Computer Systems: Mathematical Foundations," MTR-2547-I, (AD 770 768), The MITRE Corporation, Bedford, Massachusetts, March 1973.

4. D. E. Bell, "Secure Computer Systems: A Refinement of the Mathematical Model," MTR 2547-III, (AD 780 528), The MITRE Corporation, Bedford, Massachusetts, December 1973.

5. D. E. Bell, "Secure Computer Systems: A Network Interpretation," Proceedings of the Second Aerospace Computer Security Conference, McLean Virginia, December 1986, pp. 32-39.

6. K.J. Biba, "Integrity Considerations for Secure Computer Systems," MTR-3153, The MITRE Corporation, June 1975; ESD-TR-76-372, April 1977.

7. D.D. Clark, and D.R. Wilson, "A Comparison of Commercial and Military Computer Security Policies," Proceedings of the 1987 Symposium on Security and Privacy, April 1987.

8. D.D. Clark, and D.R. Wilson, "Evolution of A Model for Computer Security," Proceedings of the 11th National Computer Security Conference: A Postscript, 17-20 October 1988.

9. Department of Defense, Department of Defense Trusted Computer System Evaluation Criteria, DOD 5200.28-STD, December 1985.

10. Department of Defense, Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria, NCSC-TG-005, Version -1, 31 July 1987.

11. Department of Defense, Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria, NCSC-TG-021, Version-1, April 1991.

12. S.S. Lam, et al, "Applying a Theory of Modules and Interfaces to Security Verification," Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy, May 20-22 1991.

13. L. J. La Padula and D. E. Bell, "Secure Computer Systems: A Mathematical Model," MTR 2547-II, (AD 771 543), The MITRE Corporation, Bedford, Massachusetts, May 1973.

14. Saltzer, J.H. and Schroeder, M.D., "The Protection of Information in Computer Systems," Proceedings of the IEEE, September 1975

15. W. R. Shockley and R. R. Schell, "TCB Subsets for Incremental Evaluation," Proceedings of the Third Aerospace Computer Security Conference, Orlando, Florida, December 7-11, 1987, pp. 131-139.