APPENDIX II

RATIONALE FOR TCB SUBSETS

INTRODUCTION

A wide variety of system architectures have been proposed for the implementation of Trusted Database Management Systems, including distributed approaches, approaches requiring little or no trust in the DBMS software, approaches including nearly the whole DBMS within the TCB, and various combinations of these. It is therefore of importance to provide an interpretation for Trusted DBMS implementations that describes basic principles which are sufficiently well-grounded to support the evaluation of complex architectures. The task undertaken in this appendix is to provide such a conceptual framework.

For the simpler system architectures, it might be feasible to simply consider the Trusted DBMS as containing both trusted and non-trusted components, the trusted component to be evaluated against the Criteria and existing interpretations as a single Trusted Computing Base. A TCB evaluated as a single trusted component against the Criteria will be called a "monolithic TCB" in this Appendix. The term is not meant in a pejorative sense: it should always be valid to choose to evaluate a system in this way, and for simple systems this might be the strategy of choice.

However, systems are not always simple: for example, a system might consist of DBMS-oriented backends containing trusted components linked to one or more multilevel hosts, each with its own general-purpose TCB. Such a system might be said to be architecturally complex: it is an instance of a distributed system with responsibilities for policy enforcement allocated to different distributed components. As another example, the functional interface to the trusted component might be required to enforce a complex mixture of mandatory and discretionary policies on a multitude of object types (e.g., files, directories, relations, individual tuples, and views). Such a system might be said to be complex in the dimension of policy: the several policies to be enforced are interpreted for different objects in different ways.

For complex trusted systems, a monolithic evaluation strategy may not be attractive, economically or technically. In fact, monolithic evaluations of complex systems can be quite difficult, given that many of the design assurance techniques for higher evaluation classes are difficult to apply to large systems. It is for this reason that we must look for technically sound approaches that may be applied in lieu of monolithic evaluation strategies.

In dealing with the design and evaluation of complex system architectures, an attractive strategy is one of "divide and conquer." The presumption made hereafter is that the system is to be divided according to some strategy for its design or evaluation, the parts evaluated individually, and the composition of those parts then further evaluated for correctness. The principles upon which the the division, individual evaluations, and recomposition of the parts are to proceed must be consistent with the need to arrive at a final technical evaluation of the complete system that is technically sound.

Two distinct strategies for dealing with complex systems will be discussed here: a strategy for conducting a "partitioned evaluation," and, in contrast to it, a strategy for conducting an "incremental evaluation." A combination of these strategies can also be applied.

The partitioned evaluation strategy is appropriate for trusted systems with a complex architecture (such as one consisting of multiple, loosely-coupled processing components) but wherein the system's components may be responsible for enforcing a relatively simple access policy. Examples of such systems include networks and distributed systems. The partitioned evaluation strategy is examined in some detail in the Trusted Network Interpretation (TNI)[9] (particularly Appendices A and B) and will not be discussed in great detail here, other than to note that these principles can be applied directly to distributed architectures for Trusted Database Systems. The logical steps in designing or evaluating a system using the partitioned evaluation strategy are as follows:

1) identify the components that are the candidates for separate evaluation;

2) allocate the system's policy to the identified components by identifying the component's subjects, objects, and policy;

3) evaluate each component for its correctness in enforcing its policy and in its conformance to interface specifications; and

4) evaluate the overall system to show

a) that the architecture of the components meets the technical requirements necessary for a partitioned evaluation, and

b) that the overall system policy is enforced by the collective enforcement by the components of their respective policies.

The partitioned evaluation approach is useful because a sound design using loosely-coupled components may make some policy enforcement within individual components degenerate. For example, a physically separate component that is single-level relative to mandatory access controls need not contain a mandatory access enforcement mechanism inasmuch as access by its subjects to objects of greater sensitivity is physically precluded.

The incremental evaluation strategy is appropriate for trusted systems that are responsible for enforcing a complex access control policy, but have relatively simple architectures. The logical steps in designing or evaluating a system using the incremental evaluation strategy are as follows:

1) identify the the candidates for separate evaluation -- the candidates will be software (and possibly hardware), isolated within a contiguous set of protection domains;

2) allocate the system's policy to the identified candidates by identifying the candidates' subjects, objects, and policy;

3) evaluate each candidate for its correctness in enforcing its policy and in its conformance to interface specifications; and

4) evaluate the overall system to show

a) that the architecture of the candidates meets the technical requirements necessary for an incremental evaluation, and

b) that the overall system policy is enforced by the collective enforcement by the candidates of their respective policies.

An important goal of the incremental evaluation strategy is to attain a valid incremental evaluation for each particular TCB subset. A particular incremental evaluation for one subset is meaningful only if the more privileged TCB subsets can also be successfully evaluated. If they can be, the chain of incremental evaluations, together with a correct implementation of the domain mechanism, serves as an adequate basis for asserting that the desired composite policy is enforced. If such a chain of incremental evaluations exist, an additional TCB subset (perhaps supplied by a new vendor) and its refinement to the underlying policy can be added to the TCB at the cost of performing one incremental evaluation to extend the chain. The need for assurance that the added subset cannot invalidate the existing chain of incremental evaluations is what induces the requirement that each TCB subset reside in a separate domain.

A particular trusted system may be complex in both the architectural and policy dimensions. As one example, a system with policy-enforcing database functions allocated to loosely-coupled backend processors might be encountered. As another example, a trusted database might be implemented as a "virtual machine" available to the users of a general-purpose trusted processing environment. The partitioned and incremental evaluation strategies are compatible, and can be combined in such cases. The later sections of this appendix describe such combined strategies more fully.

The remainder examines the concept of TCB subsets by first giving an abstract description and a formal definition. Issues that require special attention are then addressed. The Appendix goes on to explain how the notion of a TCB subset and TCB partitions as defined in the TNI can be combined into the same system. Finally, a collection of illustrative examples are given.

As the principles formulated in this appendix are to be generally applicable to systems targeted for all evaluation classes, it is convenient to assume that the most stringent architectural requirements (i.e., those for Class A1) are to be met. The particular subsets of these interpretations which might be suitable for other evaluation classes will not be discussed. The ramifications of these ideas that might be suitable for other evaluation classes are discussed in the interpretations of those classes.

TCB SUBSET ABSTRACTION

A TCB subset, at its interface, is an abstract mechanism that enforces a set of policies, including some access control policy, upon subjects which attempt to access data repositories (objects) under its control. The stated access control policy must satisfy three technical conditions if the complete system (including the untrusted subjects) is to be regarded as trusted, relative to the stated policy:

- It must mediate every access.
- It must be tamperproof.
- It must be well enough structured to be evaluated for correctness.

The similarity between this and the usual definition of a reference monitor is obvious. The primary difference intended is that a TCB subset may have an internal interface to a more privileged mechanism, which is also a TCB subset, enforcing a less restrictive policy.

It is adequate for the purposes of this appendix to characterize a TCB subset in terms of the set of subjects (active entities) to be controlled, the set of objects (passive data repositories) to be protected, and the rules concerning the access of subjects to objects to be enforced by that TCB subset. For the sake of simplicity, this set of rules will be called the "access control policy" enforced by the TCB subset, although they really represent not the policy itself, (which is framed in terms of users and information), but an interpretation of a policy for application to the subjects and objects of an automated system. Arbitrary access control policies are admitted in the discussion below so that arguments can be made which are valid for any access control policy.

It is supposed that the basic security-relevant operation available to subjects is a request to a TCB subset to access a particular object in a particular access mode (e.g., search, modify, or append) implemented by that TCB subset. In response to such a request, the TCB subset may either grant or deny access.

In order to decide whether a particular request for access is to be granted or denied, the TCB subset must make a decision as to whether the requested access is consistent with the policy to be enforced. For example, an access control policy may be thought of as a list P of ordered triples <s, o, m>, where s is a particular subject, o a particular object, and m a particular access mode. Each triple denotes an access that is prohibited. The access control policy is represented as a list of prohibited accesses because failure to grant an allowed access is not normally considered an insecurity.

DEFINITION of TCB SUBSETS

A TCB subset M is a set of software (and possibly hardware) that mediates access of a set S of subjects to a set O of objects on the basis of a stated access control policy P and satisfies the properties:

1) M mediates every access to O by subjects in S;

2) M is tamperproof; and

3) M is small enough to be subject to analysis and test the completeness of which can be assured.

M makes use of the resources provided by an explicit set of more privileged TCB subsets in creating the objects of O, in creating and managing its data structures, and in enforcing the policy P; if there are no TCB subsets more privileged than M, then M uses hardware resources to instantiate its objects, to create and manage its own data structures, and to enforce its policy.

There are two additional requirements on the objects provided by the set of TCB subsets to each other and non-TCB subjects imposed for TCB subsetting to be used. The first requirement is that a TCB can be divided into TCB subsets for evaluation provided that for

any two TCB subsets M(n) and M(j), the sets of objects O(n) and O(j) are disjoint (i.e., have no elements in common). The second requirement is that a TCB subset M(j) must not provide any individual object in O(j) to more than one less privileged TCB subset.

NOTES ON THE DEFINITION

The definition of TCB subsets provided above has been formulated in a general sense without reference to the evaluation classes in the TCSEC. This approach parallels the basic definitions of Reference Monitor and Reference Validation Mechanism found in the Anderson report[2]. As was the case in the TCSEC, the allocation of these general requirements to the separate evaluation classes strengthens with the higher evaluation classes. A B2 system is required to satisfy (1) mediation and (2) tamperproofness; a B3 or A1 system is required to satisfy (1) mediation, (2) tamperproofness, and (3) smallness.

The requirement for disjoint sets of objects O(n) and O(j) is consistent with the B2 (and higher) requirements for data hiding and layering and satisfies the principle that the interface description of the system's TCB with respect to policy will not reflect the TCB subsetting (if any). It is imposed to facilitate the demonstration (for less privileged TCB subsets) of property (1) mediation of every access.

The requirement for a TCB subset to provide each particular object to no more than one less-privileged TCB subset is also imposed to facilitate the demonstration of property (1) mediation of every access. In particular, it prohibits a TCB subset structure where an object provided to a TCB subset M(j) and used for the storage of P(j)-critical information might be accessed through an alternate path of TCB subsets.

COHERENT SUBSET ARCHITECTURE

Building and evaluating a system that is intended to satisfy the definition of TCB subset in the section above requires four steps, in addition to the demonstration that the candidate TCB subsets satisfy the properties appropriate to the evaluation target class. These four steps are referred to here as the showing that the system has a "coherent subset architecture" and are the following:

- identification of candidate TCB subsets;
- "allocation" of the system policy to the candidate TCB subsets;
- an explicit description of the TCB subset structure, or architecture; and
- substantiation of the assertion that the TCB subsets occupy distinct "subset-domains".

These steps are described in the paragraphs below.

Candidate TCB Subsets

The initial step is to identify the sections of software that are candidates for being the "TCB subsets" of the system in question. The software itself needs to be clearly identified, along with the subjects and objects. In terms of the section DEFINITION OF TCB

SUBSETS, this step is the identification of M(i). S(i), and O(i). This initial identification is the basis for constructing or judging a system made up of TCB subsets.

Policy Allocation

The next step is referred to as "policy allocation". The result of this step is a description of the P(i) (which could be termed "technical policies") for the identified M(i) together with a relation of these policies to the policy of the full system P. Inasmuch as P is derived from rules or regulations and relates access of people to information whereas the policies P(i) will be expressed in terms of subjects in S(i) and objects in O(i) (e.g., defining the triples <s, o, m>, or equivalent, of the section TCB SUBSET ABSTRACTION above), the satisfaction of the TCSEC requirement that the TCB enforce its stated policy P will require that every requirement in P be traceable through the structure of the candidate TCB subsets to the TCB subset where that enforcement occurs.

TCB Subset Structure

The TCB subsets will exhibit a structure based on the ordering that is implied by dependency. TCB-subset-1 is less privileged than TCB-subset-2 if TCB-subset-1 directly uses the resources (objects, operations) provided by TCB-subsets-2, or if there is a sequence of TCB subsets from TCB-subset-1 to TCB-subset-2, where each element of the sequence directly uses the resources of the succeeding TCB subset in the sequence. In this case we use the phrase "TCB-subset-2 is more privileged than TCB-subset-1" interchangeably.

The proposed structure of TCB subsets will have to be identified. It is required that the ordering be a partial order. (footnote: Without a partial order, there would be circular chains of dependencies that would inhibit separate evaluations of the TCB subsets.)

Separate Subset-Domains

The candidate TCB subsets must be demonstrated to operate in near- isolation from each other, with the only interaction between them being that explicitly asserted as part of the interface between them. In the case of reference monitors, many existing implementations have relied on the domain concept that supports the assertions of non-bypass-ability and self-protection. For TCB subsets, a natural extension of the domain concept will be required for the isolation of TCB subsets from each other and from non-TCB entities.

Domains within a trusted system will be partially ordered. Frequently, they will be linearly ordered, particularly in implementations using ring brackets. A subset- domain is a set of domains for which (a) there is a most-privileged domain SUP; (b) there is a least-privileged domain INF; and (c) every domain between SUP and INF is included.

Each candidate TCB subset must occupy a distinct subset-domain, identified as a set of system domains satisfying the three properties of the definition in the paragraph above. Further, every domain in the subset-domain of a more privileged TCB subset must dominate every domain in the subset-domain of a less-privileged TCB subset. Lastly, any domain occupied by non-TCB software must be a less privileged domain than every domain in every subset-domain of every TCB subset in the TCB.

These requirements ensure that the structure of subset-domains (with respect to privilege) is consonant with the structure of the underlying domains and that the domains in a single subset-domain are dense and not interleaved with domains occupied either with other TCB subsets or non-TCB software.

TECHNICAL ISSUES WITH TCB SUBSETS

The introduction of TCB subsets as an evaluation strategy raises a number of technical issues that must be properly dealt with by the designers of a subsetted system if application of the strategy is to be successful. This section addresses some of these issues, focusing upon those that might be found the most troublesome or the most subtle. The topics addressed are not viewed as intractable nor as serious impediments to the construction and evaluation of a subsetted system; they are viewed in a cautionary fashion. The system builder or evaluator should be aware of the issues in order to deal with them carefully in reaching a successful conclusion. The topics addressed below are (1) the term "object" in the context of TCB subsets; (2) the issue of a TCB subset exporting objects provided by a more-privileged subset; (3) the related issues of disjoint objects and TCB subset self-protection; (4) the issue of unbypassability of TCB subsets; and (5) the issue of functional correctness of more-privileged TCB subsets as a precondition for the successful evaluation of a particular TCB subset.

Objects in the Context of TCB Subsets

As explained in Appendix I, the concept of an "object" is meaningful only if the object is related to a particular interface. Where there is only one interface relevant to a discussion (e.g., the TCB interface where a monolithic TCB is being considered), it is understood that "objects" are relative to that single interface. However, where a number of interfaces are germane to a discussion(as for TCB subsets), care must always be taken to make it clear, when the term "object" is used, which particular interface is exporting the object.

Exportation of Objects

One should note that it is always possible for a TCB subset to implement the semantics of any object type (including ostensibly directly-accessible objects). In particular, a TCB subset may re-implement objects of an object type provided to it. Typically, such a re-implementation consists simply of "passing through" the objects and operations made available to a TCB subset to the TCB subset interface. Intuitively, such objects are the "same" at each interface. Abstractly, the fact that such objects are actually "the same" is an implementation detail. The TCB subset will be said to export an object type when it provides, at its interface, objects of the same, or similar types as one of the object types provided to it.

It is in this context that the prerequisite that all the object exported by different TCB subsets must be distinct, is to be understood. What is important is that each subset must export a well-defined set of objects that it — and only it — manages. This is called the complete encapsulation of all objects. Note that directly accessible objects may be encapsulated by a TCB subset, and as an implementation detail may be exported as directly-accessible objects.

Disjoint Objects and Self-Protection

As part of the definition of TCB subsets, it was required that the sets of objects provided by the various TCB subsets be disjoint: that is, there is no object that is exported by more than one TCB subset. The purpose of this requirement is to ensure that each TCB subset mediates every access of subjects external to the TCB subset to objects under its control (i.e., that the subset cannot be bypassed with respect to the set of objects it manages). It may be difficult to see how this abstract requirement is to be reconciled to a typical design choices in the case of TCB subsets allocated to hierarchically-ordered subset domains. This subsection addresses this subtle but solvable design issue.

A subset may export directly-accessible objects. That is, it may import segments, say, from a more-privileged subset and also make segments available (as an object type) at its interface. Further, it may use segments internally to store (potentially security-critical) information about its state. Obviously, it would be a flaw if a less-privileged subset could access those segments used to store the data belonging to the TCB subset under consideration. The TCB subset is responsible for providing or correctly utilizing some mechanism allowing it to differentiate between those segments it has reserved for its own use, and those it has "exported" to a less- privileged subset. In general, although the domain structure provides a general capability for a TCB subset to protect itself from tampering, the TCB subset itself must be examined to ensure that it properly protects itself.

Non-bypassability

A more subtle set of issues than those raised by issues of self-protection arise because it is required that each TCB subset be non-bypassable with respect to the information (not just the objects) it protects. Although the objects provided at the various TCB subset interfaces are distinct, the information contained in them may be the same. Unless care is taken, it may be possible for a subject to gain access to information in an object in O(n), ostensibly protected by TCB subset M(n), by gaining access to a different object in O(m) provided by subset M(m) that happens to contain the same (identical) actual information. The issue of bypassability is of particular concern for partitioned TCB subsets (i.e., neither depends on the other) where subjects exist that can access objects of either subset.

Such bypasses are avoided by requiring, for each TCB subset, that each particular object provided at the interface be provided to one, and only one, less-privileged subset. This requirement does not preclude a subset from providing objects to more than on less-privileged subset, but ensures that no objects are shared by less- privileged subsets. The less-privileged subset is thus assured then has the choice of either (1) reserving the object for its own use, (e.g., to implement a new type of object it provides), or (2) exporting it. The only domains in which information in the object can be accessed, because it is provided to only one subset, is by accessing the object itself or some object created from it by a strictly less- privileged subset. In either case, the subset importing the object has a chance to mediate access to it and can thus control access to the information in it.

It should be noted that, as for protection, the TCB subset itself is responsible for ensuring that sharing of the objects it provides to other TCB subsets is precluded, although the domain enforcement mechanism might provide significant support for this.

Functional Correctness

One issue that arises for TCB subsets results from the observation that the evaluation of a less-privileged TCB subset depends upon the functional correctness of more-privileged TCB subsets, not simply on their ability to enforce a security policy. Abstractly, the less-privileged subset uses objects made available to is as repositories for the security critical information upon which it bases its decisions. If these repositories do not have a well-defined semantics that is correctly implemented, the validity of an incremental evaluation of the subset depending upon their behavior is cast in doubt. (One example --perhaps the most important -- of a well-defined semantics of the behavior of an object is that information placed in the object when it is last written is not modified before it is next read.)

Some perspective on this issue is gained by observing that the identical issue would arise if the combined TCB were to be evaluated as a monolith. If the objects in question (now objects exported from one internal layer of the TCB to another) function incorrectly, the TCB will be flawed and may not enforce the policy defined for it. In the monolithic case, the correct functioning is assured by engineering examination of the actual TCB source code and internal design documents available. In the context of an incremental evaluation, the correctness of a more-privileged TCB subset may, and almost certainly must, impact correctness of a TCB subset depending upon it, yet the source code and detailed interval design documentation needed to establish correct behavior is not, presumably, available to the evaluators of the less-privileged subset.

The scenario is less alarming than might at first appear. It is either the case that a successful incremental evaluation of the more-privileged subset has already occurred, or will occur (as a precondition to the successful evaluation of the entire TCB). While the functional correctness of the exported objects is not ostensibly addressed by the evaluation of the more-privileged subset, in fact the evaluation, as a side effect, necessarily produces a great degree of confidence that the exported objects function correctly: a degree of confidence that increases for the higher evaluation classes. (At class A1, for example, an FTLS has been examined repeatedly in detail and the code corresponding to each constraint explicitly identified.) In the context of a monolithic evaluation, such tasks are regarded as providing sufficient confidence that the object implementation is functionally correct. It seems a reasonable position to regard the same tasks as providing confidence that objects exported by a TCB subset are adequate to support additional subsets at the same evaluation class or lower.

TCB PARTITIONS

In the previous section, the cases where several TCB subsets were candidates for the next evaluation was deferred. The simplest such case (where multiple TCB subsets can be independently evaluated immediately) is treated in the TNI, and will be referred to here as a "partitioned evaluation strategy." The more complicated case, mixing incremental and partitioned strategies, is discussed in the next section.

Suppose that there exist two or more TCB subsets which could be evaluated immediately: i.e., which directly manage and control the physical resources and storage objects of the system and enforce identifiable access control policies upon the access of subjects to the objects under their control.

It may be observed that under the assumptions stated (that independent evaluations are possible and that each TCB subset enforces an access control policy) that there cannot exist objects which are under the control of more than one of the TCB subsets being considered. Suppose this were the case: then, it might be possible for a subject external to both TCB subsets to access the object via one TCB subset, bypassing the other. This contradicts the assumption that each TCB subset mediates all accesses to objects under its control. This argument may appear implausible, until it is understood precisely what is being said: it is certainly possible to design a system in which an object is shared between two otherwise isolated protection domains, but such a system cannot be evaluated using a divide and conquer evaluation strategy (divide and conquer): rather, the software in the two domains must be evaluated as a single TCB subset.

Because the sets of objects managed by one of these TCB subsets (and thus, any abstract objects which it exports) are disjoint, under the assumption that independent evaluations are valid the system is composed of disjoint protection domains, each of which is tamperproof with respect to the others. It follows that any subjects in the system are confined to just one of these domains).

Thus, the assumption that multiple independently evaluatable TCB subsets can be found leads to the conclusion that they are independently evaluatable precisely because the objects under their control can be partitioned into rigorously disjoint protection domains. This is the case upon which the interpretations of the TNI are based, and a system which consists of multiple TCB subsets, each of which is tamperproof with respect to the others, is called (following the TNI terminology) a Network TCB (NTCB). Each of the TCB subsets is called a TCB partition, and the disjoint set of objects and subjects comprising the domain managed by a TCB partition, and its local subjects, is called a system component.

COMPOSITION OF PARTITIONED AND INCREMENTAL EVALUATION STRATEGIES

In this section, the more complex cases where incremental and partitioned evaluation strategies may be mixed are investigated. Two modes of composition of these strategies can be identified. First, an NTCB partition within a real component of a network or distributed system may, itself, be designed and evaluated as a collection of TCB subsets within hierarchical protection domains local to the particular component containing the NTCB partition. Secondly, a more privileged TCB subset may impose a "virtually partitioned" structure upon the next less privileged hierarchical protection domain.

Subsetting an NTCB Partition

Considering an individual component of a trusted loosely-coupled network or distributed system as a trusted component in its own right (as permitted by the partitioned evaluation strategy described in Appendices A and B of the TNI), the independent evaluation of this component, based upon the policy allocated to that component, may itself be conducted incrementally, provided the component architecture provides multiple, hierarchical protection domains. The crucial issue in understanding how such an evaluation

is to proceed is to understand how the access control policy for the entire system is to be "allocated" to the component, and then to the individual TCB subsets within the component.

As previously described, the system access control policy (really a policy interpretation) may be viewed abstractly as a list of prohibited accesses <subject, object, mode>. When allocated to a particular loosely-coupled component, there is, associated with that component, a well-defined set of objects which are accessible only within that component. Thus, the access control policy as allocated to the component consists of the overall policy, with any prohibited access triples containing objects or subjects within any other component removed. The removed triples need not be prohibited locally to the component under consideration, because no local subject in that component can access a non-local object, and no local object can be accessed by a non-local subject. Thus, the NTCB partition need enforce no access control policy relative to non-local subjects, objects, or both. Thus, the local policy, i.e., the access control policy as allocated to the NTCB partition, is in some sense "smaller" than the overall global policy.

The local policy may now be subsetted and the NTCB partition evaluated incrementally as a chain of TCB subsets, provided that the needed hierarchical protection domain architecture is available. Such a strategy might be useful, for example, for a distributed implementation of a trusted DBMS, allowing, for instance, a trusted front-end controller to be evaluated independently of trusted database engine back-ends. Either the controller or the back-ends or both might be further subsetted (e.g., into TCB subsets enforcing mandatory and discretionary policies) as part of the partitioned evaluation of the component.

Virtual Partitions

Another approach for composing the partitioned and incremental evaluation strategies is to have a more privileged TCB subset create a "virtual partitioning" of the next less-privileged TCB subset. The basic technique for doing this is to enforce a policy at the interface below the virtually partitioned system which partitions all of the objects exported at the interface into disjoint sets. A subject executing in the next less- privileged hierarchical domain would then have access to objects in one, and only one, of these mutually exclusive sets. The virtual partitioning can clearly be created by means of access controls which assign to each external subject and object a label representing the virtual partition to which they belong, and prohibiting all access by subjects in one virtual partition access to objects in another, (in addition to whatever other access control policy is being enforced). Such a "virtual partition" is, in fact, just an isolated protection domain. If this approach is taken, the incremental evaluation of the next less privileged TCB subset can be performed as a partitioned evaluation of each of the virtual partitions separately, using the TNI guidelines to compose the independent partition evaluations.

Once a hierarchical protection domain has been divided into mutually exclusive virtual partitions (isolated domains), the question arises whether any TCB subsets in less privileged hierarchical domains need to be similarly partitioned. Surprisingly, this does not need to be done. Both abstract and concrete arguments lead to this result.

Abstractly, a subject which can access objects in different virtual partitions of an underlying hierarchical domains can do so only if both accesses are permitted by the respective partitions, in accordance with their allocated policy subsets. Since the enforcement of these policies by the virtual partitions has been evaluated to be correct and cannot be invalidated by the activity of external subjects, the accesses are, by definition, allowed by the global policy. One might observe that if the virtual partitions are viewed as independent virtual machines, subjects external to them which can communicate with multiple virtual machines are, in effect, secure connections between the virtual machines.

More concretely, the transfer of data from one virtual partition to another is always mediated by the most privileged TCB subset, because it must take place ultimately using either storage objects or devices. Thus, the transfer of information does not actually bypass the more privileged TCB subsets, as the more abstract view might suggest.

The use of virtual partitioning finds an immediate application in the trusted DBMS environment as a way to implement a trusted DBMS which operates in a "stand-alongside" mode with a trusted operating system. The most privileged TCB subset might, in addition to a basic access control policy (such as a mandatory policy) provide for two isolated protection domains in the next less privileged hierarchical domain. Within these domains a general-purpose TCB enforcing a discretionary access control policy upon the objects it controls, in the other, a more specialized trusted DBMS enforcing discretionary access controls upon the objects it exports. A subject in the application domain, which is less privileged than either the DBMS or the general-purpose TCB, could freely access either DBMS or TCB objects in a way that is consistent with the enforced policies.

The two basic methods for combining the incremental and partitioned evaluation approaches may be applied recursively to analyze systems which are even more complex. For instance, in the example above, the DBMS virtual partition might be subsetted into TCB subsets which enforce discretionary controls upon basic DBMS objects (e.g., real relations) and, in a less-privileged hierarchical subdomain, additional discretionary controls upon more complex objects (e.g., views). The subsetting of the DBMS virtual partition need not be paralleled within the general TCB partition, as they are independently evaluatable virtual partitions. The complete evaluation of the resulting system would proceed as follows:

— Evaluate the underlying Basic TCB for enforcement of mandatory access controls, as well as provision of the required domain structure.

— Evaluate the general-purpose TCB incrementally for the correct enforcement of discretionary controls in the environment provided for it by the Basic TCB. (As this is an incremental step, the Basic TCB evaluation must be completed before the general-purpose TCB virtual partition can be evaluated.

— Evaluate the more privileged DBMS domain for the correct enforcement of discretionary controls upon its exported objects (real relations). This is an incremental step and logically cannot be performed before the Basic TCB evaluation is complete.

— Evaluate the less privileged DBMS domain incrementally for the correctness of its

enforcement of discretionary controls on views.

POLICIES, MODELS, AND SPECIFICATIONS

The requirements in the TCSEC for showing how the various representations of the system in question fit together can be represented as in Figure II-1. In the monolithic case, the Policy must be enunciated; the Model must be prepared and maintained and be shown to be sufficient to enforce the policy; the DTLS (and FTLS for A1) has to be constructed and be shown both to correspond to the model and to be corresponded to by the TCB implementation, as represented in its interface specifications and its source-code implementation.

That set of requirements allows a chain of reasoning to proceed from the stated policy that the system in question is intended to enforce down to the actual source code of the TCB.

In the case of TCB subsets, the intent must remain the same. At any of these levels of discourse, it must be possible to be able to trace a feature or quality or characteristic both to the left on the diagram or to the right.

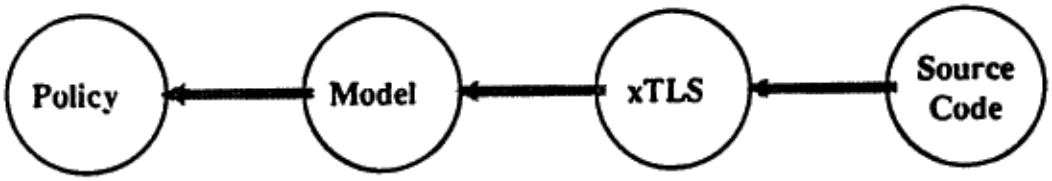
The dilemma is that the research community has not yet established either (1) how to tightly or formally subdivide a model or a Top Level Specification (either DTLS or FTLS) into a set of models or xTLS for the individual TCB subsets, or (2) how to tightly or formally "compose" a set of models (respectively, xTLS's) into a "unified" or "composed" model (respectively, xTLS). However, while a rigorous methodology for these compositions might be valuable in increasing the overall assurance of a subsetted TCB beyond Class A1, the non-availability of such techniques does not preclude the evaluation of a subsetted TCB using a strategy of incremental evaluation, with each TCB subset evaluated as a monolith executing upon the virtual machine on which it depends.

The position adopted for this set of interpretations is that, while designers and evaluators must explicitly and directly face the problem of showing how the policies, models, and TLSs for individual TCB subsets fit together as descriptions of a TCB enforcing the overall unified policy, informal arguments are acceptable evidence that a coherent security architecture has been achieved. This position is considered appropriate for systems targeted up to Class A1, given the current non-availability of more rigorous composition methods. This approach recognizes that arguments concerning the composability of domain-based TCB subsets, although not currently rigorously formalized, are sufficiently well-grounded to support the evaluation of highly assured, subsetted TCBs. The nature of the informal arguments to be presented is discussed below, in conjunction with Figure II-1.

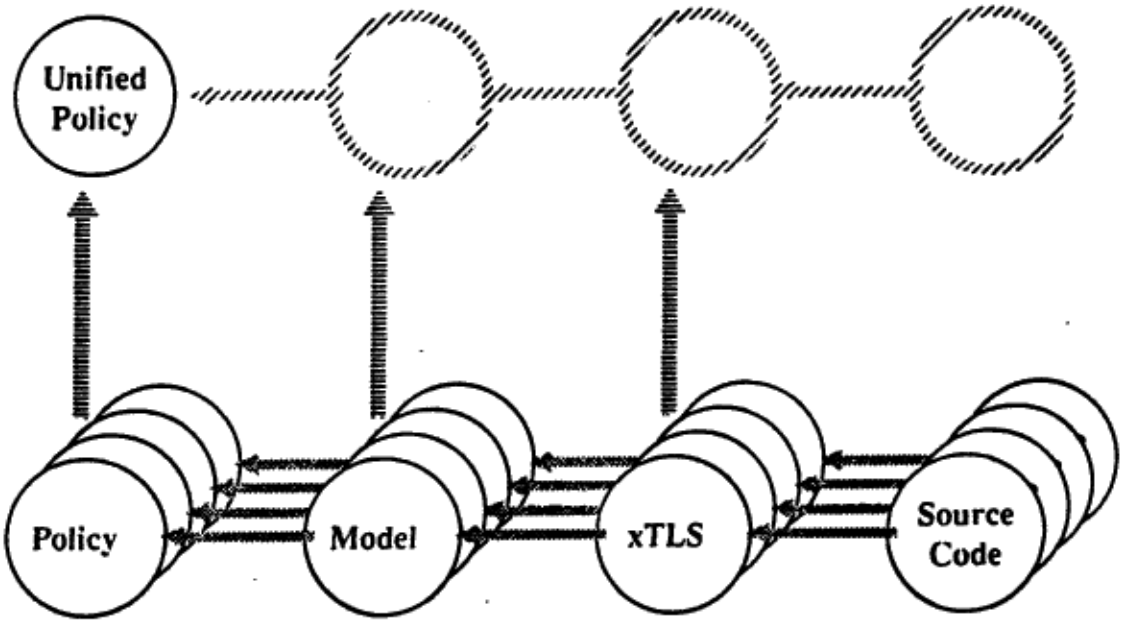
Hence the requirement for establishing that a TCB subset situation pertains includes several crucial topics. The first topic is a policy allocation to the subsets that is a precise division of the policy for the entire system. The task set in the demonstration of "consistent" implementation of the system's policy is to support the patent assertion that the policy division (type of policy and the objects and subjects to which it applies) does not blur across TCB Subset boundaries.

The second topic is relating the models of separate TCB Subsets to the entire system. The requirement in this Interpretation is that each TCB subset meet all of the requirements of

Figure II.1
Consistency Chains



a. Monolithic TCB Case.



b. TCB Subsets Case

the Monolithic TCB case with respect to its allocated policy and its own model, xTLS, and source code. That done, there is added the requirement that the sufficiency of the set of models to represent abstractly the Unified Policy of the entire system must be described explicitly and that the description will be convincing. The convincing argument that the set of models work together to represent the entire system abstractly is indicated by the arrow pointed upwards from the set of Models in Figure II-1(b).

The third topic, the efficacy of a set of DTLS's (in the case of A1, DTLS's and FTLS's) to specify the TCB interface with respect to exceptions, error and effects, is treated in a similar fashion: an explicit and convincing description of how the xTLS available for each TCB subset work together is added to the requirement. Failure to meet of all these issues satisfactorily will result in the rejection of the assertion that the system meets the requirements for TCB Subsets.

Satisfactory completion of these requirements is logically equivalent to demonstrating, for each step of the evaluation, that a unified model for the entire TCB is consistent with the policy, that a unified TLS corresponds to the model, and so on, just as for a monolithic evaluation of the entire TCB, as illustrated in Figure II-1(b). Rather than formally composing the policies, policy models, specifications, and so on and performing a single monolithic evaluation using the resulting evidence, a series of individual incremental evaluations is performed (one for each TCB subset). The evaluations are then tied together by presenting the required informal arguments that the individual policies enforced collectively subsume the required unified policies, that the individual models collectively represent the allocated policies, that the individual specifications represent appropriate unified interfaces, and that the TCB subsets' source code is consistent with their xTLS's. As an illustrative example, the security architecture for the complex example just presented will be reviewed.

The overall access control policy for the example consists of mandatory and discretionary subsets. The mandatory subset has been allocated to the Basic TCB, and the discretionary subset to the next less privileged hierarchical domain. The union of the allocated policies must be no less stringent than the original, combined policy: if so, the TCB subsets in the first two hierarchical domains collectively, if they are correct, will enforce the desired overall access control policy.

The discretionary policy is further allocated to the general TCB and DBMS partitions within the second hierarchical domain. The allocation is rather straightforward as the sets of objects protected by each are exhaustive and disjoint: the policy is enforced either by the general TCB or the DBMS depending upon which virtual partition the object is exported by. What is important, from the evaluation point of view, is that the underlying real policy (i.e., that expressed in terms of the discretionary access of users to information) is essentially the same for each of the partitions: the interpretation of this policy in terms of either general TCB or DBMS objects may differ (as they are objects of a different kind), and the particular policy models may differ as required to support their independent, incremental evaluation. These models need not mention mandatory controls at all, as enforcement of mandatory controls, in this example, is the responsibility of the underlying basic TCB, and the hierarchical protection architecture and prior evaluation of the basic TCB ensures that the mandatory policy is enforced for subjects in any less privileged domain.

Within the DBMS virtual partition, the allocated DAC policy must be correctly subsetted into that for real relations (enforced by the more privileged DBMS domain) and that on views (enforced by the less privileged DBMS domain). When composed, the conclusion that discretionary controls upon both real relations and views by the entire DBMS partition is valid. Finally, the conclusion that the actual discretionary policy is always enforced either by the general TCB or the DBMS is also valid.

Just as different models may be used (although the formalisms in each model must accurately represent the allocated policy), the formal or informal top-level specifications need not be combined. Each must represent a complete, and accurate description of the interface presented by the trusted subset within the assigned protection domain, and must be properly mapped to the corresponding model.

The TCSEC provides Criteria for the evaluation of Trusted Computer Systems that are framed in terms of a monolithic TCB, i.e., a TCB that consists of a single TCB subset. In interpreting these Criteria for TCBs containing multiple TCB subsets, it is convenient to introduce technical terms for frequently recurring concepts so that the interpretations and rationale can be made concise.

The term "composite TCB" means the complete collection of all TCB subsets. An important special case is that where the TCB consists of a single TCB subset. Where the phrase "the TCB" occurs in the interpretations, the composite TCB is meant. The term "TCB subset" is used when the emphasis is on an evaluatable TCB subset that is part of a larger composite TCB.

The term "TCB subset" is meant to encompass both hierarchical and non-hierarchical TCB subsets: that is, a particular TCB subset may, or may not, be more privileged than some other TCB subset, and it may not be possible to compare two given TCB subsets with regard to privilege. Where the emphasis is upon the non-comparability of two TCB subsets, the term "TCB partition" may be used.

In order to avoid confusion, spatial metaphors for the relation between two TCB subsets have been avoided. The terms "more privileged" and "less privileged" are preferred to the less precise "inner," "outer," "higher," "lower," etc. The notion that one TCB subset is "contained in" another has been rigorously avoided: a TCB subset may be "less privileged" than another, but it consists of distinct code and data from the TCB subset it depends on.

The term "less privileged subject," when used in the context of a particular TCB subset, should be understood to mean any subjects to which the interface presented by the TCB subset is accessible. These may include both non-TCB subjects, and subjects of a less privileged TCB subset, depending upon the TCB subset architecture.

The "most privileged TCB subset" (in each partition) is that with direct access to the hardware and storage media resources (of the partition). The most privileged TCB subset corresponds to the "reference validation mechanism" described in section 6.3 of the Criteria.

The "least privileged TCB subsets" are those that present a direct interface to non-TCB subjects. The case should be noted where both of two TCB subsets have such an interface, even though one is strictly more privileged than the other (i.e., the less privileged TCB subset does not mediate some calls to the more privileged TCB subset). Some of the Criteria relating to "the TCB interface" have special interpretations for any TCB subsets included in the set of "least privileged TCB subsets" (including both of the TCB subsets in the case described above).

Of course, in the monolithic case, the single TCB subset is identical to the composite TCB, and is the single "most privileged" and "least privileged" TCB subset at one and the same time.

Many of the Criteria (such as those for covert channel analysis) are specifically concerned with the enforcement of a mandatory security policy. In interpreting the Criteria for a TCB composed of multiple TCB subsets, it is convenient to be able to precisely distinguish those TCB subsets to which the interpreted Criteria applies. Other Criteria have different interpretations depending upon whether a TCB subset is depended upon (directly or indirectly) for the enforcement of the mandatory security policy, or not. The term "mandatory TCB " is introduced to distinguish this class of TCB subsets.

The "mandatory TCB" is that set of TCB subsets, designated by the designers of the TCB, subject to the following properties:

- Any TCB subset to which any component of the mandatory policy is allocated for enforcement, must be within the mandatory TCB.
- Any TCB subset responsible for security functions that directly supports the mandatory policy (such as designation of the current level of a single-level device), must be within the mandatory TCB. It is specifically noted that audit functions do not directly support the mandatory policy, since they provide no access control.
- For any TCB subset within the mandatory TCB, all TCB subsets more privileged than that TCB subset must also be within the mandatory TCB.

Determination of whether a particular security-related function supports the mandatory policy requires considerable care. Audit need not be included within the mandatory TCB because it is considered to augment, not support, the mandatory policy enforcement. Identification and authentication functions typically must be performed within the mandatory TCB, because most systems include single-level devices (e.g., terminals) whose current level is set based upon the clearance of the current user. However, it is possible to describe systems that do not depend upon the identity of a user to establish the level of the device: for example, all terminals might be placed in physical locations where the clearance of the user for each terminal is implicitly associated with the accessibility of the terminal. For such systems, the identification and authentication function does not support the mandatory policy, and these functions need not be performed within the mandatory TCB.

The definition of mandatory TCB given above was deliberately framed to permit the designers of a system to include TCB subsets within the mandatory TCB that could be omitted from it without violating the above constraints. This design choice is considered compliant with the Criteria, even though it appears to violate the principle of least privilege, because the same system could be evaluated against the Criteria directly as a monolithic TCB. System designers intending to use the TCB subset approach are strongly encouraged, however, to consider designating a "minimal" mandatory TCB.

In the context of balanced assurance (see Appendix IV) the notion of the mandatory TCB becomes extremely important, as a composite TCB may be minimally compliant with the Criteria for Classes B2 or above by meeting the assurance criteria for the Class within the mandatory TCB, and the assurance criteria for Class C2 for all other TCB subsets. It should be noted that a system to which the balanced assurance concept is applied must be designed as a system of multiple TCB subsets (balanced assurance cannot be applied to monolithic systems). It should also be noted that the TCB subset concept is valuable in its own right (even where balanced assurance is not employed) as a means of exploiting the principle of least

privilege to reduce the difficulty, and cost, of a complex composite TCB.

SUBSETTED M-TCBS AND TRUSTED SUBJECTS

One technique for augmenting TCBs enforcing a mandatory security policy that is frequently encountered in practice is the use of so-called "trusted subjects" that implement new objects and enforce the underlying mandatory security policy for those objects. The Multics mail system described in Appendix I is an example of the use of this technique. This design technique, although it typically offers substantial reductions in effort for the re-evaluation of the augmented TCB, is not an instance of TCB subsetting. The term "trusted subject" is used in the precise sense defined for the Bell and LaPadula model (Bell, 1974), viz., a subject in a less-privileged domain than that of a TCB component required to enforce a mandatory security policy, that is allowed by the component to "write down" in sensitivity, but is otherwise constrained by the more-privileged TCB subset to obey the mandatory security policy. The use of such an approach often significantly reduces the total technical effort involved in evaluating the trusted system, because it is only necessary to show that the trusted subject does not permit unauthorized information flow; an easier condition to verify than that it is functionally correct (i.e., implements exactly its specifications).

However, the two protection domains cannot be considered independent TCB subsets, because the more-privileged component cannot, in general, be shown to enforce he policy subset allocated to it as it does not include a component (the trusted subject) that has the capability to violate the policy. Abstractly, the trusted subject must be included with the more-privileged TCB component to form a single TCB subset; they cannot be independently evaluated. This abstract conclusion is matched by engineering experience; demonstrating that a trusted subject is flow-free typically requires access to technical data concerning the more-privileged TCB component. Similarly, the introduction of additional trusted subjects into a TCB subset, in general, requires the re-evaluation of that subset. The designer of a TCB subset may choose to support trusted subjects. Multi-domain TCB subsets can exist so that within a TCB subset the designer may reserve a protection domain (or domains) for the use of trusted subjects.

Many proposed architectures for trusted DBMSs have been based upon the use of trusted subjects to manage labeled storage objects of higher granularity (e.g., tuples) that are stored in labeled objects (e.g., segments) exported from a more-privileged domain along the lines of the Multics mail system as summarized in Appendix I. Such designs cannot be evaluated as independent TCB subsets but must receive a single, monolithic evaluation for the reasons explained above. It should not be concluded, however, that mandatory policy subsets cannot be allocated successfully to true TCB subsets. Two brief examples are given.

> If a designer can successfully meet the challenge of supporting new storage objects by allocating them to properly-labeled storage objects supplied by an underlying TCB subset enforcing a mandatory security policy using subjects not trusted with respect to that policy, a new TCB subset enforcing the same mandatory policy on the new objects has been created. Examples of paper designs using this approach are the architectures proposed by Hinke and Schaefer, Bonyun, and the SeaView

project. One advantage of adopting such an approach is the use of TCB subsets to allow the database component to receive an incremental evaluation.

A second example would be the implementation of a mandatory policy subset (e.g., support for categories) that refines the mandatory policy subset enforced by a more privileged subset (e.g., only hierarchical sensitivity levels).

ILLUSTRATIVE EXAMPLES WITH DBMS APPLICATIONS

EXAMPLE 1: HIERARCHICAL TCB SUBSETS

This example constitutes a design for a B2 secure database management system based on a B2 secure operating system providing protected process domains and facilities for communicating between these domains.

The DBMS vendor has designed a product that works in conjunction with an operating system product, which has already been evaluated at the B2 class. This DBMS product can, therefore, be evaluated using the incremental evaluation approach. Using this approach, it is not necessary to re-evaluate the B2 operating system. In fact, the evaluation report from the operating system evaluation should be used as input to the evaluation of the DBMS.

This example describes a TCB that is comprised of two TCB subsets: an operating system and a DBMS. The operating system TCB spans two levels of privilege and is the m-TCB. The DBMS TCB executes in a third privileged domain and enforces a discretionary security policy on database objects.

The Operating System

The operating system product was not evaluated using a TCB subset approach, but by using a monolithic approach. Therefore, the entire operating system TCB is identified as a single TCB subset, in this case, the most privileged subset spanning two domains. The more privileged domain provides process domain isolation, maintains integrity over the labels of storage objects, which it also maintains, and performs mandatory access control between processes (subjects) and storage objects. It also provides primitive services, including interprocess communication services and I/O device control services. The interprocess communication services, which are subject to mandatory access control checks, are the only means by which one process can affect another process. The storage objects include processes, devices, interprocess communication buffers, and segments.

The standard operating system supports a file subsystem. Although the file subsystem is included in the operating system TCB subset, it operates at the next lower level of privilege than the operating system. It uses I/O device control services and segment handling primitives, provided by the operating system, to provide a secure multilevel file structure. The file subsystem provides a new set of encapsulated objects, files, which it maps into segments. Each file is associated with exactly one security level. In addition, the file subsystem provides a set of trusted functions that can be used to manipulate these encapsulated objects: create, delete, open, close, read, and write. The file subsystem also implements a discretionary access control policy over the encapsulated objects that it presents at its interface.

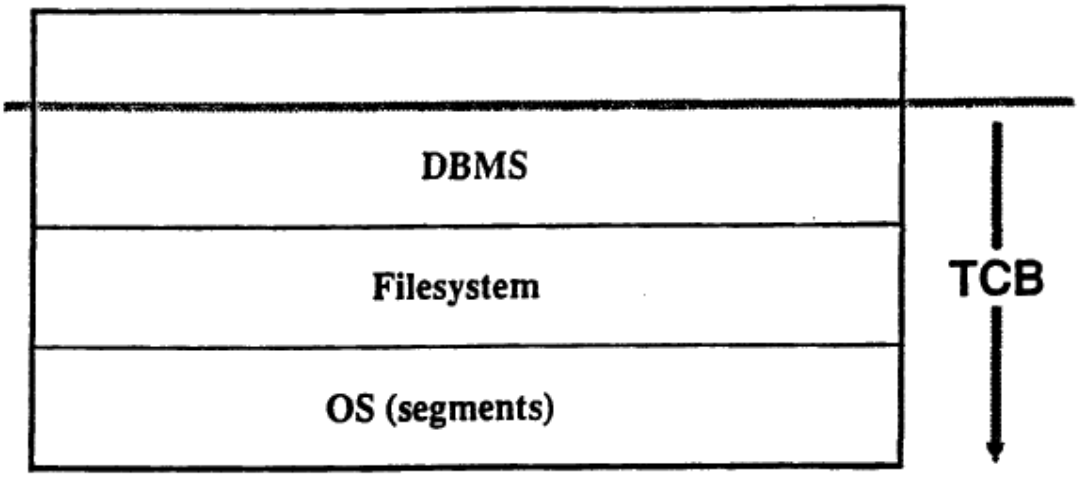
Database Management System TCB Subset

In this example, the DBMS is separated into two parts: the DBMS frontend is untrusted code that operates at the security level of the user's process. Most of the parsing, query processing, and other semantic processing is performed by the untrusted DBMS frontend. The DBMS backend is a TCB subset that resides at the next lower level of privilege from the file

subsystem. From this hierarchical position, the DBMS TCB subset can use any of the services provided by the file subsystem as well as any of the more primitive operating system services that are not hidden by the file subsystem. This architecture allows the DBMS TCB subset to use the mandatory access controls provided by the operating system by mapping database objects to files, where the labels are affixed and mandatory accessed control is enforced. The DBMS TCB subset provides a new set of named objects (e.g., elements, rows, columns, tables, views) and the appropriate services for manipulating those objects. The DBMS TCB subset enforces a discretionary security policy on these named objects and is outside the mandatory TCB.

Figure II-2 illustrates the hierarchical TCB subset approach discussed above. An untrusted DBMS front-end process is established by the operating system TCB subset for each DBMS user. In the course of the DBMS session, the user's request for data is processed by the DBMS frontend and translated into requests for named database objects (e.g., rows) provided by the DBMS TCB subset. The requests are sent (using operating system interprocess communication services) to the DBMS TCB subset for processing. These are translated into file system requests which are forwarded (using operating system interprocess communication services) to the file subsystem. The operating system retrieves the requested data using its I/O device control services. The file subsystem builds a buffer for each encapsulated object and sends it back to the DBMS TCB subset. The DBMS TCB subset processes the file data and builds a new buffer for each of its named objects. These interprocess communication buffers are then sent to the untrusted DBMS frontend process using operating system services.

Figure II.2
Less Privileged DBMS Backend TCB



EXAMPLE 2: INTEGRITY LOCK

This is included as a contrasting example of something that is not a TCB subset. This example is built on the same secure operating system as in the Hierarchical TCB Subsets example. In that example, an operating system TCB subset and a DBMS TCB subset were introduced. It will be useful to the reader to understand the functions of these TCB subsets. The reader is also referred to the Guideline on Integrity Lock Architecture for a more complete discussion of the Integrity Lock Approach.

The Segmented DBMS

The DBMS in the Hierarchical TCB Subsets example was implemented in two parts: a trusted DBMS backend process, responsible for access to the physical database and an untrusted DBMS frontend process, responsible for parsing, query processing, and other semantic processing. In the Integrity Lock example, the functions provided by these two processes are the same as in the Hierarchical example, but neither of the processes is trusted. Instead the untrusted DBMS backend process (HIGH) executes in a domain at the system high security level and the untrusted DBMS frontend process (LOW) executes in a domain at the security level specified by the user. A trusted sanitization filter TCB subset (the Integrity Lock filter) is provided by the DBMS vendor to perform the following actions: 1) to take objects provided to it by HIGH and sanitize and downgrade these objects to the same level as LOW; and 2) to take objects provided to it by LOW, to associate the appropriate security indicator with them using a cryptographic checksum, and to pass these objects, their security labels, and their checksums, to the HIGH process for storage into a database maintained at the system high security level.

In this example, the operating system assures that only the appropriate process (i.e., the HIGH process) is permitted to directly access the files that contain the system high DBMS. Similarly, the trusted sanitization filter must be invoked as part of each communication between the LOW and HIGH portions of the DBMS.

The trusted sanitization filter in this architecture is not a TCB subset because it does not enforce a security policy. Rather, it should be considered as a trusted process with the mission of downgrading the sensitivity of portions of storage objects. For a more detailed explanation, see the Guideline on Integrity Lock Architecture.

EXAMPLE 3: MONOLITHIC TCBS

This architecture refers to computer systems where the trusted portions of the system (i.e., the TCB) are amalgamated into one logical partition. The entire TCB may be comprised of different software modules, each of which performs a particular function, but the entire TCB executes in one logical domain on the hardware base. A logical domain is defined by the processor state on some hardware and by the ring mechanism on other hardware. Whatever the mechanism for defining the domains, the TCB will execute in the most privileged domain. Of course, all of this implies that the entire TCB is contained on a single hardware base, and systems that are comprised of multiple hardware bases do not fit the monolithic TCB concept.

Unlike hierarchical TCBs, the monolithic can not be incrementally evaluated. As a single TCB that is self-contained in one domain, the hierarchy of privileges that is required for incremental evaluations is not present.

DBMS - Operating System Coexistence

Consider a general purpose operating system that is manufactured by Company A for use with its mainframe computer system. The operating system is evaluated and rated by the NCSC. Company A does not manufacture a DBMS to be marketed with its operating system, so the operating system evaluation does not take any DBMS into consideration. During the evaluation, the TCB has been precisely specified to include the operating system only.

Now, Company B markets a DBMS that is designed for Company A's operating system and hardware base. Company B asks the NCSC to evaluate their DBMS as it is used with Company A's products.

Upon examination of this combination of products, the evaluators find that the DBMS alters the original TCB provided by the operating system by invading the protection domain in which the operating system executes. There are several reasons why the DBMS might do this. On reason is to enhance performance by augmenting certain operating system services which are not efficient for the DBMSs purposes. Another reason is so the DBMS can protect itself from tampering by subjects external to the TCB. The former has been discussed at length in a paper by Stonebraker[26]. The latter assumes that there is only one protected domain on the hardware base in which security-relevant software can be placed.

Since the original TCB has been altered by addition of the DBMS, the evaluation of the original operating system TCB is no longer valid. Thus the result of adding the DBMS to the operating system is a different TCB, which must be evaluated as if the original operating system had never been evaluated. No assumptions can be made that the operating system contains the same security functionalities and assurances that were found to be true in the original evaluation. Contrast this with the evaluation of hierarchical TCB subsets, where the evaluation evidence of more privileged TCB subsets can be utilized in the evaluation of the DBMS.

When evaluating a DBMS/operating system combination as a monolithic TCB, the evaluation must examine the overall security policy enforced by the combination and the assurances that the combined TCB is tamperproof and non-circumventable. For example, the security policy enforced on the database by the DBMS must not be in conflict with the security policy of the operating system. Users who are prohibited by the DBMS from accessing objects in the database must not be able to gain access to those same objects by going through the operating system. Likewise, all audit files and other protection critical data maintained by the combined TCB must not be accessible to unauthorized subjects either through the DBMS or through the operating system. Security testing must be performed using a realistic configuration of the combined TCB. After the evaluation is complete, a new rating will be assigned to the combined TCB. This rating may be lower than, equal to or higher than the rating of the original TCB.

EXAMPLE 4: PARTITIONED TCB

A partitioned TCB is a special case of a TCB, which is composed of subsets, but the subsets are not ordered hierarchically. Instead, the TCB subsets are ordered in vertical partitions. Each partition is confined to its own domain by either logical separation on the same hardware base or physical separation on different hardware bases. An example of partitions separated logically on the same hardware base was given in the discussion of hierarchical TCBs. An example of physically separated partitions is a database machine and front-end configuration where part of the TCB is on the database machine and part is on the front-end computer.

As an example of a physically partitioned TCB, consider a TCB consisting of two TCB subsets. One of the TCB subsets is a general purpose operating system TCB that enforces a security policy over a specific set of objects (e.g., files). The other TCB subset is a database machine. The database machine enforces a security policy on databases, which are completely separate and disjoint from the files protected by the operating system TCB subset.

The operating system TCB subset and the DBMS TCB subset can each be evaluated in isolation. These evaluations are conducted in much the same manner as the evaluation of network partitions, as is described in the Trusted Network Interpretation. The combination of the TCB subsets must also be in accordance with the composition rules in Appendix A of the TNI. Thus, the evaluation of a database machine against the TDI is but one step in the evaluation of a physically partitioned system.

Database machines can be built either as a monolithic TCB or they can use a hierarchical TCB subset architecture. It depends on the internal structuring of the database machine's TCB as to whether it can be evaluated using the incremental method or as a monolithic TCB. Evaluations of database machines are complicated somewhat by the necessity to connect them to a front-end computer. A great deal depends on whether the database machine has all of the necessary functions for a complete TCB. If it does not, some security relevant functions must be implemented on the front-end, and the TCB becomes analogous to a network TCB (NTCB). The reader should refer to the Trusted Network Interpretation, Appendices A and B for more information on evaluating NTCBs.

Database machines that do contain all of the necessary functions for a complete TCB can be evaluated in isolation. Either incremental or monolithic evaluation methods can be applied to the database machine evaluation, whichever is appropriate to the architecture of the database machine.