87-3098

TCB SUBSETS FOR INCREMENTAL EVALUATION

William R. Shockley, Roger R. Schell
Gemini Computers, Inc.

Abstract

In this paper, the notion of a TCB subset is defined and applied to the problem of evaluating a complex trusted system. The general strategy proposed for a complex evaluation is one of "divide and conquer": the TCB is partitioned into disjoint TCB subsets, each of which resides in its own protection domain and enforced a security policy subset upon the subjects and objects under its control. The following protection domain structures (as ordered by privilege) are considered: purely hierarchical (the "protection ring" case), purely isolated (the "distributed system case"), and hybrid cases. The basic architectural and policy constraints induced by such structures are presented and a few illustrative applications of the ideas described.

1. Introduction

The growing availability of trusted computing base (TCB) evaluated products has produced significant interest in how such products may be incorporated within more complex application systems for eventual field deployment. Typical examples of systems which have been proposed and designed include trusted distributed systems, trusted networks, trusted database management systems, and trusted systems incorporating within one system a variety of these approaches. A crucial subproblem which must be solved if such complex systems are to be assembled with a high degree of assurance highly-assured way is to develop techniques whereby complex systems can be evaluated for their conformance to the technical criteria provided by the Department of Defense Trusted Computer System Evaluation Criteria (the TCSEC) [1].

In dealing with any complex system, an attractive strategy is one of "divide and conquer". What is wanted is an approach for dividing the trusted component of the system into simpler parts, evaluating each of the parts separately, and then validating the correctness of the way the parts are composed to form a single system enforcing a globally well-defined security policy. It is obviously desirable that the validation of the correct composition of evaluated parts be a relatively simple task. The simplicity of the reference monitor abstraction, which is the guiding theoretical abstraction underlying the TCSEC, lends itself well to a "divide and conquer" evaluation strategy. In this paper, two distinct strategies will be described, compared, and finally combined within one unifying conceptual framework. This analysis may be considered an extension of the principles first described in [2], with the benefit of several additional years of experience and thought in how a complex TCB might be composed from a collection of simpler TCBs residing in individual protection domains.

The first, a strategy for the "partitioned evaluation" of a trusted distributed system or network, depends upon the notion that a complex system may be decomposed into independent, loosely-coupled, inter-communicating processing components. This strategy is most suitable for trusted systems which have a complex physical architecture (e.g., distributed systems or networks). The partitioned evaluation strategy, as described in this paper, is a restatement of the principles presented in Appendices A and B of the recently-distributed Trusted Network Interpretation of the TCSEC, (the TNI) [3], which provides interpretations for the application of the TCSEC criteria to networks.

The second strategy, which is in this paper is denoted a strategy for the "incremental evaluation" of a trusted computer base, depends upon the idea that a complex TCB may be divided into simpler TCB subsets, each of which enforces a subset of the global access control policy. The effect of such a policy decomposition, when allocated to the various TCB subsets, is to allow for a chain of simpler evaluations (each called an "incremental evaluation") to be performed, leading to an overall conclusion that the composite TCB is correct. Unlike a partitioned trusted system, incrementally evaluated TCB subsets are thought of as residing in the same, tightly-coupled processing component. The incremental strategy is particularly well-suited for trusted systems which enforce relatively complex access control policies over virtual objects which are very different from the physical storage objects provided by the processor hardware. For that reason, the TCB subset abstraction is currently receiving substantial attention as a key notion upon which interpretation of the TCSEC for trusted DBMS implementations (the Trusted Database Interpretation, or TDI) [4] might be based.

The partitioned and incremental evaluation strategies are compatible, and may be combined in various ways for trusted systems which are complex both in the architectural and policy dimensions. Two distinct ways in which the strategies may be combined will be presented later in the paper after the "pure" versions of the approaches have been characterized.

The remainder of this paper is organized as follows. Section 2 provides an abstract description of a "TCB subset" which is responsible, in general, for enforcing an arbitrary policy for the access by subjects (active entities) to objects (passive data repositories) under its control. A TCB subset is a generalization of the reference monitor abstraction, but differs from it in that one TCB subset may (but need not) include a simpler TCB subset from which it obtains services. Section 3 defines the notion of incrementally evaluated TCB subsets precisely and characterizes what it means for a TCB subset in such a group to be correct. Section 4 recapitulates the notion of a partitioned TCB evaluation drawn from the TNI and shows how a collection of network TCB partitions (as defined in the TNI) may be interpreted as a collection of independently evaluatable TCB subsets (thus unifying the two approaches under a single idea). Section 5 describes how incremental and partitioned evaluation techniques may be combined in a single system. Section 6 recapitulates and summarizes the ideas presented from the perspective of their practical application.

As the principles formulated in this paper 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. Relaxations of these which might be consistent with lower evaluation classes will not be discussed.

2. TCB Subset Abstraction

A TCB subset, at its interface, is an abstract mechanism that enforces some access control policy upon subjects which, from time to time, attempt to access data repositories (objects) under its control. Informally, it must satisfy three technical conditions if the complete system (including the untrusted subjects) is to be regarded as secure, 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 [1] is obvious: indeed, a reference monitor is a TCB subset. The primary difference intended is that a TCB subset may have an internal interface to a smaller included mechanism, which is also a TCB subset (enforcing a less restrictive policy). The term "NTCB partition" (drawn from the TNI) will be reserved for a TCB subset which does not include a smaller TCB subset, and thus is in direct control of a particular, well-defined subset of storage objects, devices, and other hardware components of one processing component of a particular trusted system. The term "reference monitor" will be reserved for the collection of NTCB partitions in a particular system. (The distinction between a "reference monitor" and "the TCB" which is normally drawn, even for monolithic systems, is not important for the discussion in this paper and will not be pursued).

In the degenerate case of a single, monolithic trusted system, (the sort of system described by the TCSEC), there is a single TCB subset which is identical to the TCB, and may be viewed as the single NTCB partition of a degenerate distributed system, as well. The TNI typically views a distributed system or network as a collection of NTCB partitions, one for each of many loosely-coupled processing components, which together form a single TCB or reference monitor. Each of these NTCB partitions is a distinct TCB subset, as defined above. The TCB architecture described in this paper consists of a collection of TCB subsets which are allocated to hierarchical protection domains: at least one example of such an architecture is currently under developmental evaluation (e.g., GEMSOS [5]). In the case of a "pure" hierarchy of TCB subsets, the collection of TCB subsets comprises a single TCB which is (degenerately), also a single NTCB partition. More complex protection architectures which mix the ideas of NTCB partitions and hierarchical TCB subsets in non-degenerate ways will be introduced toward the end of the paper.

It is adequate for the purposes of this paper to characterize a system which contains 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 the TCB subset of interest. 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 the TCB subset) to access a particular object in a particular access mode (e.g., read, write, or execute). 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 access control policy to be enforced. Abstractly, this 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) of accesses which must be prohibited. For instance, if the triple < x, myfile, read> appears in the list, subject "x" may not be given read access to object "myfile". The convention of representing the abstract access control policy as a list of prohibited accesses has been used, because failure to grant an allowed access is not normally considered an insecurity.

3. Properties Of Hierarchical TCB Subsets

In the previous section, an abstract characterization of a TCB subset mechanism enforcing an abstract access control policy (represented by a set P of prohibited accesses) was provided. In this section, the notion of a TCB subset which includes other TCB subsets will be precisely defined and abstractly characterized.

The central idea is that the set P of prohibited accesses is to be replaced by a collection of subset policies P(i), the union of which includes every element of the original set P. Corresponding to each such policy subset a distinct TCB subset M(i) is postulated to exist. It is supposed that every request for access to an object o is submitted to each TCB subset responsible for enforcing a policy with respect to o (i.e., a prohibited access to o occurs somewhere on that TCB subset's list of prohibited accesses). Access is to be granted only if every such TCB subset M(i) permits access.

Under the described circumstances the original policy P is enforced by the collection of TCB subsets. If a particular access is prohibited in P, there must exist at least one subset of P, P(i), which prohibits the access. (Otherwise, the constraint that the union of all of the P(i) includes P is not met). The particular TCB subset M(i) enforcing the subset policy P(i), when consulted, will prohibit the access. Thus, every access prohibited by P will be prohibited by the collection of TCB subsets collectively. At worst (in the case where the union of the subset policies includes P as a proper subset), the collection of TCB subsets might deny accesses which are not prohibited by P, but as previously indicated, this is not normally considered an insecurity.

It is now appropriate to ask how such a collection of TCB subsets might be organized in a real system in such a way as to ensure that every TCB subset responsible for protecting a particular object was consulted on every access to that object, and in such a way that the goals of an incremental evaluation can be properly met. The answer (concisely) is that the postulated collection of TCB subsets will be implemented within a series of hierarchical protection domains [6], with the objects to be protected by each TCB subset exported (i.e., provided as an abstraction) at the domain interface between that TCB subset and the next (or the application domain, for the least privileged TCB subset). The remainder of this section is devoted to explaining how such a protection architecture within the composite TCB can serve as a valid basis for an incremental evaluation.

An evaluation strategy for the collection of TCB subsets will be considered to be "purely incremental" if there is some ordered sequence of evaluation increments, one for each TCB subset, such that the correctness of each evaluation increment depends upon, and only upon, a positive evaluation for the previous increment (if the current increment is not the first). (This strategy may be contrasted with that for a partitioned evaluation, as for a trusted network, in which the evaluations of each component individually are essentially independent of the evaluations of other components).

Under the assumption introduced by this definition, there will be exactly one of the TCB subsets in the collection which is to be evaluated for correctness first (with regard to the policy subset it enforces), and its evaluation must not depend in any way upon the correctness of any of the others. It follows that this particular TCB subset (call it M(1)) must be privileged to directly access the real system objects and resources (i.e., memory, devices, and so on): M(i) is the system reference monitor. The correctness of the reference monitor can be independently established only if no other TCB subset can tamper with it. Thus, it must inhabit a protection domain of its own which is tamperproof with regard to all other system or application software: it is the most privileged of the collection of TCB subsets.

Once the reference monitor M(1) has been identified and independently evaluated, there will be exactly one of the remaining TCB susbsets, M(2), which is now to be evaluated. By the definition given above, M(2) depends for its correctness on the correctness of M(1), but not upon any of the other TCB subsets. (Systems for which more than one TCB subset might be evaluatable after M(1) will be characterized later). Therefore, the evaluation of M(2) can proceed by regarding the hardware, together with M(1), as a virtual machine upon which M(2) executes. In fact, M(2) may be regarded in all respects (including the criteria for its evaluation) as a reference monitor which controls the virtual resources (objects, devices, etc.) exported by M(1).

This characterization also is the guide for how to apply the TCSEC Criteria to M(2): provided a particular Criterion is relevant to the policy P(2) allocated to M(2), it may be directly applied to the evaluation of M(2), just as if M(2) were a real reference monitor executing upon a real machine, rather than a TCB subset executing upon the virtual machine provided by M(1).

Recall that in order to be able to establish the correctness of M(1), it had to be assumed that M(1) could not be compromised by any other TCB subset, including M(2). This means that M(2) must reside in a different (and less-privileged) protection domain than M(1). The incremental evaluation of M(2) requires that M(2) be tamperproof with respect to subjects external to M(1) and M(2). Thus, M(2) must reside in a protection domain that is less privileged than M(1) but more privileged than all other TCB subsets and applications.

Applying this argument recursively, a structure of hierarchical protection domains emerges. Each TCB subset occupies its own hierarchical protection domain which is tamperproof with respect to all less privileged subsets and any untrusted applications. An adequate protection architecture for supporting an incremental evaluation strategy is provided by the well-known "protection ring" structure implemented, for example, by Multics [7] or GEMSOS [6]. One may observe that it is possible to support a valid ring abstraction which is implemented in software, provided that a minimum hardware capability exists for two distinct protection domains (i.e., privileged and non-privileged), as was done for early implementations of Multics [7].

The concept of a "protected object" requires some interpretation in the context of such a collection of TCB subsets organized within hierarchical protection domains. Each successive TCB subset may be viewed at its interface as providing a collection of objects over which some composite policy (the union of the policy subsets for the TCB subsets) is enforced. These objects may be divided into two general categories: those which are inherited with no essential changes from underlying TCB subsets (although additional policy restrictions may be enforced), and those which are abstractions built from more primitive underlying objects by the intervening TCB subset.

It is the second class of objects, those which are abstractions built by the intervening TCB subset, which are often called "interpretively accessed", "indirectly accessed", "virtual", or "abstract" objects. No notion of "direct access" to such an object (external to the TCB subset implementing it) can be supported, because is not, by definition, a storage object managed by the reference monitor, M(1). "Access" to such an object is represented by the availability, at the TCB subset interface, to commands which allow the state of such objects to be modified, or observed: these commands result in the execution of programs which actually modify the internal representation of the object, or translate and copy data in this representation into objects the invoking subject can directly access.

In order to support such interpretively accessed objects, the TCB subset must store the representation of its state into objects the TCB subset can modify, but that subjects external to the TCB subset cannot. The latter requirement stems from the consequences of permitting such access: if an external subject could directly modify the state representation of an abstract object, the abstraction presented at the interface of the TCB subset could no longer be shown to be tamperproof. Moreover, if the stored states of many such abstract objects are stored in a single storage object (a typical case), external subjects cannot be allowed to directly read the storage object either, because that might give them read access to abstract objects which they are not authorized to read, in addition to the one they are allowed to read.

Thus, a particular TCB subset may be characterized functionally as inheriting a collection of objects from the next more privileged TCB subset (or from the hardware, for M(1)), removing some of them for its own use and re-exporting the rest, along with whatever new abstract objects it may implement. What is meant by saying that a TCB subset is "tamperproof" is not only that its software cannot be modified by external subjects, but that the representations of the abstract objects it implements for interpretive access must be stored in storage objects which cannot be directly modified (and often, not directly observed) by external subjects.

It should now be apparent that the hierarchical structure of protection domains required in order for a chain of incremental evaluations to be workable imposes, as well, a hierachical allocation of the totality of objects to be protected by the TCB to those domains. A particular object, exported at a particular TCB subset interface, is either inherited from a more privileged TCB subset, or exported as a virtual object implemented by the particular TCB subset in question. Ultimately, a single TCB subset can be identified for each object which is the "source" of that object: those objects inherited from the hardware and exported by the reference monitor M(1) are what are generally called the "storage objects" by the TCSEC.

The allocation of objects to their originating TCB subsets induces a constraint upon the policy subsetting described at the beginning of this section: a policy subset P(i) allocated to a particular TCB subset M(i) may include prohibitions only upon those particular objects exported by M(i) to less privileged protection domains. These objects may either be inherited from more privileged TCB subsets, or exported as new abstract objects implemented by P(i). The challenge of actually designing a system incorporating multiple, hierarchical TCB subsets is two-fold. First, the selected policy subsets must combine to represent a composite policy which is at least as restrictive as the original policy to be enforced. Second, each policy subset must be interpretable exclusively in terms of those objects and abstractions which are to be made available at the particular TCB subset interfaces in the selected order of evaluation. No individual policy subset must be represented in terms of abstract objects which will be implemented only by a less-privileged TCB subset.

As a more concrete example, suppose that a trusted DBMS is implemented as a TCB subset in a hierarchical protection domain less privileged than a conventional TCB (which provides the virtual environment for the DBMS software), but more privileged than the applications which use the DBMS. It might store relations in segments available to it from the conventional TCB and enforce a discretionary access control policy upon its application subjects upon attributes (columns) of a relation. The particular segments within which relations were stored could not be made directly accessible to applications for either read or write access, since they might then be able to read or modify attribute values to which the subject did not have authorized access. Thus, the segments used to store relations would not be objects visible at the DBMS interface: in their place would appear a collection of system function calls which provided interpretive access to relations. A function which provided indirect "observe" access to an authorized attribute value would (if closely examined) involve the copying of data by the DBMS into an object (such as a parameter area) which was directly accessible to the invoking subject.

4. NTCB Partitions

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

Suppose that there exist two or more TCB subsets which could be evaluated immediately and independently (i.e., the correctness of one evaluation does not depend upon the other). For this to be the case, the set of objects exported by one TCB subset must be disjoint from that exported by the other: otherwise, neither TCB subset could enforce a policy relative to that object independently, for such a policy could be bypassed by accessing the same object via the other TCB subset. It may well be the case that for a particular allocation of policy subsets or protected objects, this possibility might not result in an insecurity: the point is that this fact cannot be established by examining just one of the TCB subsets relative to its allocated policy. In short, in order to preserve the desired property that the TCB subsets can be independently evaluated, shared objects must be excluded from their respective interfaces.

An immediate question arises: if shared objects do not appear at the interfaces of the two TCB subsets, how may a subject utilizing the services of one TCB subset communicate with a subject controlled by the other? The solution to this problem advanced by the TNI is to view the communication between two independent TCB subsets as taking place via "communication channels" which are abstracted as collections of devices local to the communicating TCB subsets, coupled by a medium for information transfer. Implicitly, the policy subsets to be enforced locally are stated strictly in terms of local devices and objects: it is assumed that the communication medium, the only shared resource, cannot be directly accessed by subjects, and that no access control policy must be enforced by any TCB subset relative to it (more precisely, the policy to be enforced is implicit in the policy enforced for access to the local devices).

This argument may appear implausible, until it is understood precisely what is being said: it is certainly possible to design a distributed system in which an object is shared between two otherwise isolated protection domains, but such a system cannot be evaluated using the desired evaluation strategy (divide and conquer): rather, the software in the two domains must be evaluated as a single, monolithic TCB subset.

Because the sets of storage objects managed by one of these TCB subsets (and thus, any abstract objects which it exports) are disjoint, the system is composed of disjoint protection domains, each of which is isolated with respect to the others. It follows that any subjects in the system are confined to just one of these domains (because a subject is a <process, domain> pair) [1].

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 an NTCB 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.

5. Hybrid Systems Of TCB Subsets

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 will 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 under consideration. This strategy may be viewed as one (relative to that component) of partitioning, followed by incremental subsetting within that component.

Secondly, a more privileged TCB subset may impose a "virtually partitioned" structure upon the next less privileged hierarchical protection domain. This strategy may be viewed as one of incremental subsetting, followed by partitioning within a subset.

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 (this is the fundamental condition upon which a partitioned evaluation strategy is predicated). 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.
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 (partition, then subset) 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.

Partitioning a TCB Subset

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 in a hierarchy of TCB subsets. The basic technique for doing this is to control access to objects exported to the partitioned, less-privileged TCB subset in such a way that the exported objects are partitioned into mutually exclusive classes. A tamperproof attribute for each external subject executing in the next less privileged domain indicating which class of objects it may access. 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 classes of objects. Such a "virtual partition" is, in fact, just an isolated protection domain. If this approach is taken, the evaluation increment for the next privilege level can be conducted as a series of independent evaluations, one for each of the TCB subsets into which the TCB component residing at the next privilege level has been partitioned. The requirement to provide for completely isolated domains is not so much a requirement for correctness as a precondition for supporting the desire to evaluate the next layer of TCB subsets independently.

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 (in fact) 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 (mistakenly) 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 one of these domains would reside a general-purpose TCB subset enforcing a discretionary access control policy upon the objects it controls (e.g., files or segments). In the other domain would reside a more specialized trusted DBMS enforcing discretionary access controls upon the objects it exports (e.g., relations, tuples, attributes, or views). A subject in the application domain (less privileged than either the DBMS or general-purpose TCB) could freely access (consistent with the enforced policies) either DBMS or TCB objects, and, consistent with the enforced access modes, transfer data between them freely.

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 (such as real relations) and, in a less-privileged hierarchical subdomain, additional discretionary controls upon more complex objects (such as 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 underling 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.

6. Summary And Conclusions

In this section, the ideas presented in the body of the paper will be briefly summarized from the perspective of their practical application.

We have argued that a complex TCB can be divided into a set of simpler TCBs which can be independently or incrementally evaluated. Each TCB subset is allocated to one of a set of protection domains which are partially ordered by "privilege": if a domain A is more privileged that B, then a subject in A may access any object allocated to B. If, of two domains A and B, neither is more privileged than the other, they are non-comparable. Requiring that the domains be partially ordered excludes certain valid domain architectures allowing otherwise non-comparable domains to share objects. Corresponding to the allocation of TCB subsets to domains is an allocation of policy subsets to the TCB subsets. Each TCB subset can be evaluated either incrementally (i.e., as a reference monitor executing upon a virtual machine), or independently (i.e., as a reference monitor executing upon hardware) for enforcement of its allocated policy subset: the resulting system enforces a global policy representing the composition of the policy subsets. A characterization of how policy subsets are to be allocated in both the independent and incremental cases has been given.

As for any "divide and conquer" strategy, several motivations might exist for the adoption of a purely hierarchical, a purely partitioned, or a hybrid approach for the evaluation of a complex trusted system.

The first motivation is technical: as one of the fundamental requirements for the evaluation of a TCB at the higher evaluation classes (i.e., Class B3 or above) is the systematic reduction of complexity so that a high degree of assurance can be attained that the TCB is correct, any technique for decomposing a complex TCB into simpler parts which might be independently or incrementally evaluatable is valuable. This is particularly important where the composite TCB might be too complex to evaluate as a monolith. In order to support this motivation, substantial care has been taken in the formulation of the concepts presented here to retain a fundamentally sound method for composing individual TCB subsets, and the policies they enforce, into a composite TCB enforcing a composite global policy.

The second motivation is economic: in many cases, it may be that the cost of the individual TCB subset evaluations, plus the cost of validating their composition, may be less that for evaluating an equivalent monolith. This effect might occur where non-linear costs, relative to the complexity of the system (e.g., for verification of an FTLS) might exist.

The third motivation is market-oriented: the capability to consider independently evaluate partitioned NTCB components provides a means for vendors to provide specialized, trusted network or distributed system components without having to bear the entire cost or risk involved in developing and having evaluated a complete network system. Similarly, the ability to consider incrementally evaluated hierarchical TCB subsets provides a means for vendors to add value by extending the security perimeter of an appropriately designed, pre-existing TCB subset without having to bear the entire cost or risk involved in developing and sponsoring for evaluation the entire TCB. This would seem a prerequisite for the existence of evaluated, third-party trusted software.

In this paper, the notion of a TCB subset has been presented: a strategy for identifying individual modules of a TCB, each of which has the essential characteristics of a reference monitor, but enforces only a subset of the global access control policy to be enforced, and executing either in a loosely-coupled system component or on the virtual machine provided by a more privileged TCB subset. The actual architecture chosen by a system designer depends strongly upon the protection domain architectures available and the specific policy requirements which must be enforced for each component. The underlying "glue" which allows the resulting collection of TCB subsets to be viewed as a single TCB is the careful subsetting of the overall policy into policy subsets, and their allocation to individual TCB subsets.

Acknowledgements

The research reported in this paper was funded under subcontract agreement no. 87-007-1 to Infosystems Technology, Inc. in support of the National Computer Security Center's efforts to produce draft TNI and TDI documents.

The ideas presented in this paper have been continually tested and refined by the members of both the TNI and TDI working groups, whose contributions and critical review the authors gratefully acknowledge. This paper represents neither an official position of the National Computer Security Paper nor an approved consensus position by either working group: the authors bear full responsibility for any errors in interpretation of approved criteria or guidelines which may be found herein. Acknowledgement is also due to our colleagues at Gemini, whose many day-to-day contributions to the ideas presented represent much of the actual substance of the paper.

References

[1] **Department of Defense Trusted** Computer System Evaluation Criteria. Dept. of Defense, National Computer Security Center, Dec. 1985. DOD 5200.28-STD.

[2] M. Schaefer and R.R. Schell. Toward an Understanding of Extensible Architectures for Evaluated Trusted Computer System Products. In **Proc. 1984** Symp. on Security and Privacy, IEEE Computer Society, pages 41-49, 1984.

[3] **Trusted Network Interpretation of the** Trusted Computer System Evaluation Criteria. National Computer Security Center, Jul. 1987. NCSC-TG-005 Version-1.

[4] M. Hale. Status of Trusted Database Management System Interpretations. In **Proc. 10th National Computer Security** Conf., pages 340-342, 1987.

[5] R.R. Schell, T.F. Tao, and M. Heckman. Designing the GEMSOS Security Kernel for Security and Performance. In **Proc. 8th National Computer Security** Conf., pages 108-119, 1985.

[6] L.J. Shirley and R.R. Schell. Mechanism Sufficiency Validation by Assignment. In **Proc. of the 1981 Symp.** on Privacy and Security, IEEE Computer Society, pages 26-32, 1981.

[7] E.I. Organick, **The Multics System:** An Examination of Its Structure, The MIT Press, Cambridge, Mass. and London, England, 1972.