Trusted Computing Update

Marshall D. Abrams, Ph.D.
Michael V. Joyce

The MITRE Corporation
7525 Colshire Drive
McLean, VA 22102
703-883-6938
abrams@mitre.org

This is the second paper of a series of three related papers that examine contemporary information technology concepts. The first paper examined how the security community developed the accepted concepts and criteria, looked at changes currently ongoing, and provided insight into the driving forces and probable directions. This paper presents contemporary thinking about object management and summarizes vertical and horizontal extensions to the Trusted Computing Base (TCB) concept.

1 INTRODUCTION

The evolution in computer software architecture has prompted significant changes in the structure of security functions and the composition of trusted information technology systems. Conventionally, most of the security-relevant functions are concentrated within the operating system. Often, these functions, especially those dealing with access control, are commingled with object management functions. This natural inclination to commingle these functions draws attention to the relevance of object management in achieving security goals. Following an introduction of the object manager function, this paper examines the relationships between object management and the vertical and horizontal extensions for structuring the Trusted Computing Base and describes representative examples of access control techniques in contemporary information technology systems.

This paper was first published in Computers & Security, Vol. 14 No.1 pp. 57-68, © Elsevier Advanced Technology 1995, Oxford, UK; http://www.elsevier.nl/locate/compsec. This is the second of three related papers.

2 OBJECTS AND OBJECT MANAGEMENT

An object manager is a logical function responsible for handling and controlling entities in the logical address space. The operations performed by an object manager span the full life cycle of an object that includes creating objects, processing service requests, and removing objects from the logical address space. The following discussion of object management focuses on the creation of objects and processing of access requests. Although many implementation strategies are possible, for this discussion we will assume that all management responsibilities for a particular type of object are vested in one Object Manager and that management responsibilities for a type of object are not shared between Object Managers.

Creating an object is the most fundamental operation performed by an Object Manager. We view the creation of an object as a transformation. That is, an Object Manager, perhaps as a result of a series of complex operations, creates a logically addressable object from resources. This concept of a transformation of resources to objects by an object manager is fundamental to the concepts advanced in this paper. This transformation can be expressed as:

$$\text{Object Manager: } \text{Resource}_1, \ldots \text{Resource}_n \varnothing \text{ Object}$$

Given this view of the creation of an object, one wonders, "where do the resources come from?" There are many potential providers of resources. In general, an Object Manager can create a new object by combining several resources, subdividing a resource, or passing a resource without change. A file system Object Manager, for instance, might obtain a collection of disk sectors and transform them into a file. On the other hand, a memory manager might pass memory segments directly to an application.

The transformation of resources into objects is not limited to Object Managers within the operating system. In contemporary Information Technology systems, many complex applications contain Object Managers. A database management system, for example, may contain an Object Manager that creates database management system objects such as tuples and tables. An Object Manager in a message management system might manage objects called messages.

An Object Manager within an application performs the same function as an Object Manager within the operating system: it transforms resources into an object. Application Object Managers may obtain their resources from the operating system, the Bedrock, or other applications. After obtaining these resources, the application Object Manager transforms them into new objects, called application objects. The objects created by these Object

Managers are distinguished by their granularity and structure, often related to their intended use. To obtain a finer granularity, applications, such as database management systems, often subdivide a resource into application objects.

2.1 RELATIONSHIPS AMONG OBJECT MANAGERS

To begin the discussion of the relationships among Object Managers, let us examine the Object Managers in the operating system. These object managers are responsible for transforming the hardware and firmware—the bedrock on which the system is built—into objects. The hardware and firmware, for instance, might implement a virtual memory address space composed of segments. Within the operating system, an Object Manager might convert one or more of these resources into a new object, a file, or a directory that is introduced into the logical address space.

From the point of view of an Object Manager, an object is a resource that is used to construct new objects. The Object Managers within the operating system use the Bedrock resources to create new objects, such as files. When an object is made available by the operating system, it becomes a resource for an application Object Manager. That is, one Object Manager's object is another Object Manager's resource. A fundamental duality exists. In the relationship between the operating system and the application, we observe:

$$\text{Object}_{\text{OS}} + \text{Resource}_{\text{application}}$$

The relationship of the resources, Object Managers, and objects in an untrusted system is illustrated in Figure 1. In the evolution of objects, the bedrock is the starting point, the supplier of the most fundamental resources in the system. One of the Object Managers within the operating system obtains these resources and transforms them into a new object. The application views this operating system object as one of its resources, Res_{appl}. An Object Manager within the application transforms those resources into an application object.

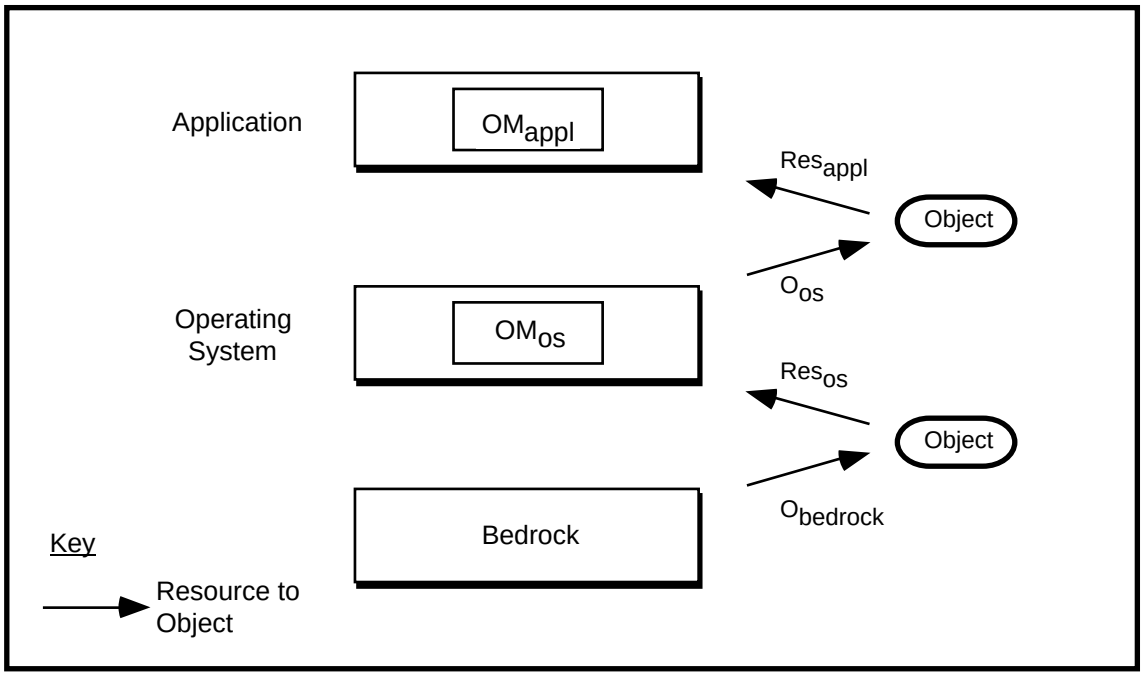

Figure 1. Objects and Resources in an Untrusted Operating System

2.2 OBJECT MANAGEMENT AND ACCESS CONTROL

In a trusted environment, besides creating an object, an object manager must address access control requirements. Creating an object makes the object accessible by other processes. To satisfy the security objective central to the purpose of trusted systems, the Object Manager must include some type of *access control function* that implements an Access Control Policy and adjudicates all access requests. The inclusion of an access control function distinguishes the Object Manager process in a trusted operating system from that in an untrusted operating system. This function adjudicates an access request based upon the relevant security attributes and the rules that implement the Access Control Policy. The result of the adjudication is a decision either to permit or to deny the access request.

Figure 2 illustrates the interrelationship of the components in the trusted environment. The principal element in the figure is one of the object managers in the operating system. The object manager is composed of both an object management function and an access control function. When an application requests access to an object, the relevant security attributes of the application and object are validated by the access control function. If the attributes compare favorably according to the rules that define the Access Control Policy, then the access is permitted.

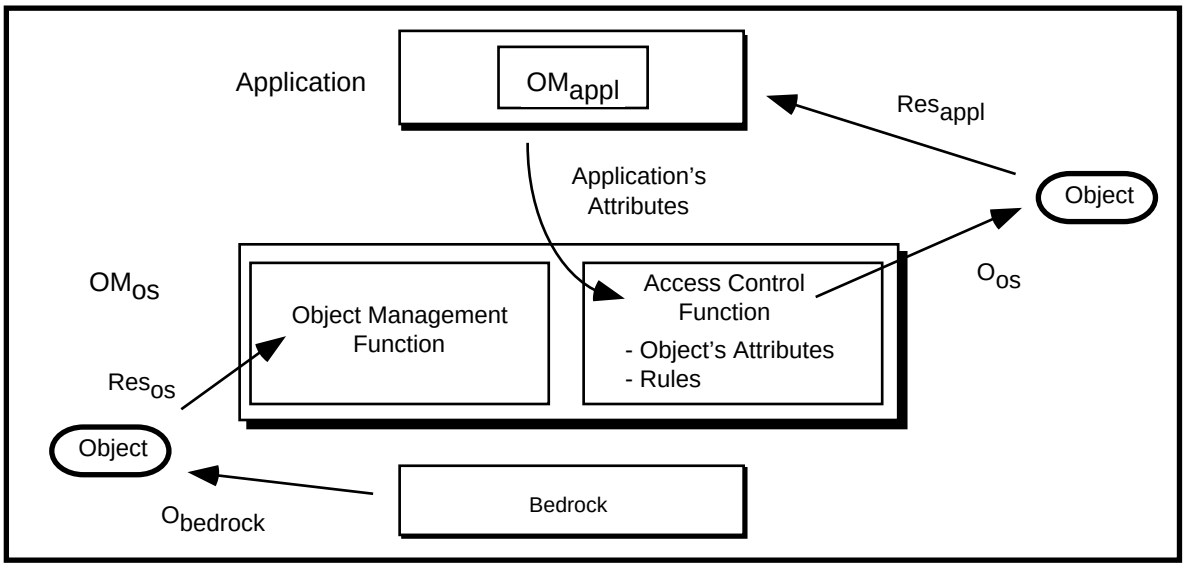

Figure 2. Object Manager in a Trusted Operating System

2.3 OBJECT MANAGEMENT IN POLICY-ENFORCING APPLICATIONS

In contemporary information systems, the concept of object managers has moved beyond the boundary of the operating system and is performed by Policy-Enforcing Applications. In general, a Policy-Enforcing Application contains object management and access control functions similar to those found in an operating system. The Object Manager in a Policy-Enforcing Application transforms the resources it obtains into one or more new objects. As with an operating system, the transformation might involve combining several resources, subdividing a single resource, or passing a resource unchanged. The effect of the transformation is that a new object (O_{PEA}) is created and is potentially available to any other application. When an application attempts to access one of these objects, the Policy-Enforcing Application must adjudicate the request and make a decision either to permit or to deny the access request. Hence, in a trusted environment, a Policy-Enforcing Application includes an access control function that implements its Access Control Policy.

Figure 3 illustrates the relationship of the object management and access control functions in a Policy-Enforcing Application. The functions and the relationship of those functions are similar to those seen in a trusted operating system. The Object Manager obtains resources

and transforms them into objects. An application can access these objects only after the access control function in the Policy-Enforcing Application validates the request. The access control decision is based on a combination of access control rules and access control attributes associated with the application and the object.

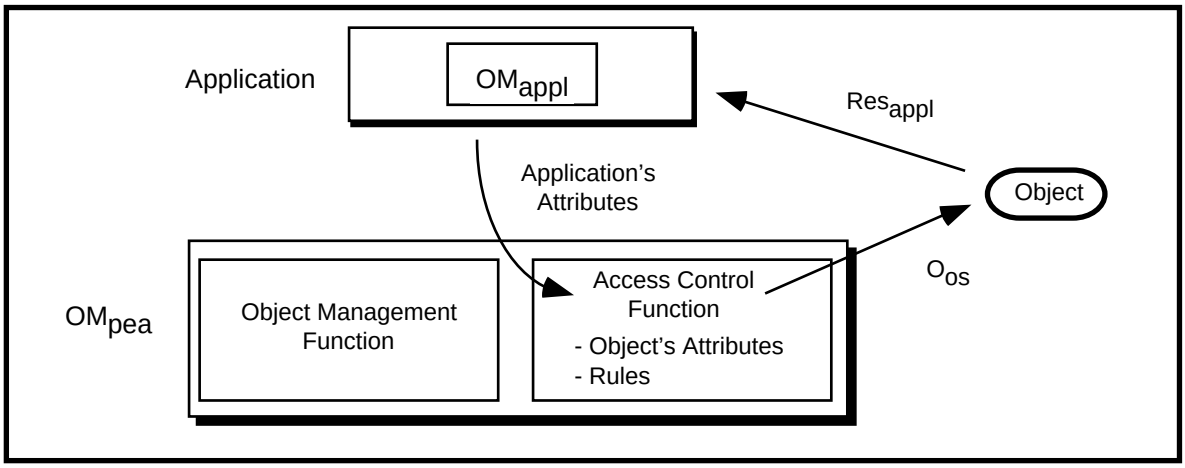

Figure 3. Object Management and Access Control in a Policy-Enforcing Application

3 TRADITIONAL TRUSTED COMPUTING BASE CONCEPTS

The understandings of trusted technology has continued to evolve since they were formalized nearly 20 years ago. The early understandings were that the hardware and software that had to be trusted generally equated to the operating system and the supporting hardware. As Policy-Enforcing Applications were introduced into the system architecture, trusted was required in additional components, not just the operating system. This section examines the evolution of trusted concepts and how they address the needs of complex system architectures.

3.1 EVOLUTION OF CONCEPTS

The first paper in this series introduced several fundamental security concepts. The Anderson Report (Anderson, 1972) introduced the Reference Monitor concept as an ideal to

achieve controlled sharing. As defined in the Anderson Report: "The function of the Reference Monitor is to validate all references (e.g., references to programs, data, peripherals) made by programs in execution against those authorized for the subject (e.g., the user). The Reference Monitor not only is responsible for assuring that the references are authorized to access shared resource objects but also to assure that the reference is the right kind (i.e., read, or read and write, etc.)." A combination of hardware, software, and firmware that implements the Reference Monitor concept is called the *Reference Validation Mechanism*..

The understanding of architecture for trustworthy computing continued to expand from this beginning. When early prototypes were built, the Reference Validation Mechanism proved insufficient. The set of hardware and software that had to be trust was extended. This new set was called the Trusted Computing Base. A Trusted Computing Base includes not only the Reference Validation Mechanism but also encompasses all other functionality that directly or indirectly affects the correct operation of the Reference Validation Mechanism. Administrative and auditing mechanisms are examples of the types of supporting functionality that do not make access control decisions but are placed within the Trusted Computing Base boundary.

Using an operating system as an example, figure 4 illustrates a synthesis of object management concept and the Trusted Computing Base notion. An operating system contains a cross-section of capabilities necessary for the efficient management of a set of resources. Some of these capabilities have security relevance. The two functions that are the focus of our attention, the object management and access control functions, have obvious security implications. The object management function is responsible for creating objects, processing service requests, and removing objects from the logical address space. The access control function contains both the rules and the security attributes that support the access control decision-making process. Together, these functions encompasses all functionality that directly or indirectly affects the correct operation of the Reference Validation Mechanism and constitute the Trusted Computing Base. The Trusted Computing Base is represented as the high-lighted portion of the operating system in figure 4.

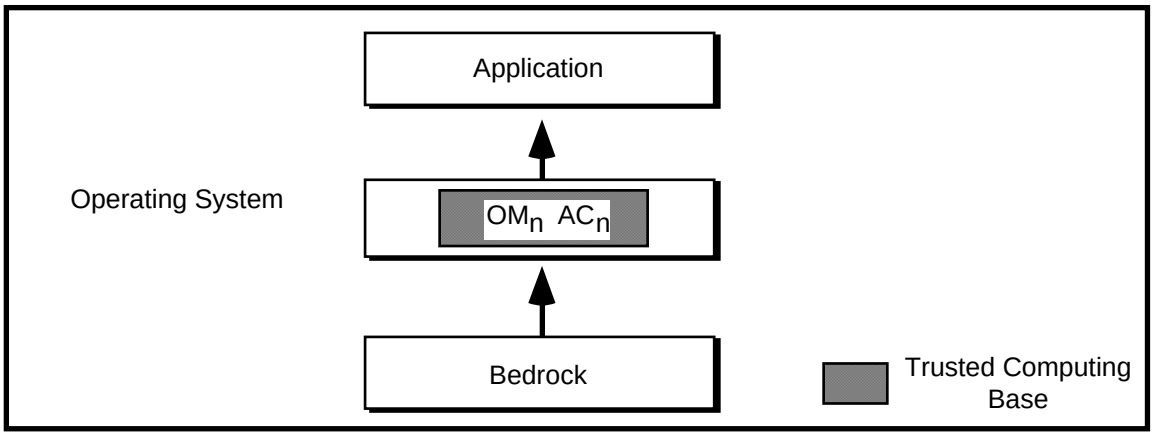

Figure 4. Composition of a Trusted Computing Base

3.2 NON-SUBSETTED TRUSTED COMPUTING BASE

The Orange Book describes the traditional non-subsetted Trusted Computing Base. The Orange Book provides system developers and system acquirers with a well-defined set of criteria to measure the security provided by a computer system. The criteria, which are grouped into six hierarchical classes, define the features and assurances considered necessary to ensure the protection of information in a computer system in a Department of Defense environment. The criteria within the system features group include a security policy. In the C division, there is only one policy, Discretionary Access Control; in the A and B divisions, both Mandatory Access Control and Discretionary Access Control are implemented.

The understood architectural framework contained in the Orange Book is an operating system containing one or more object managers and access control functions. In figure 5, we illustrate a conventional implementation of the Mandatory Access Control and Discretionary Access Control mechanisms in a non-subsetted Trusted Computing Base. Although only one Object Manager is included in the figure, typically several object managers would be present in an operating system. The mechanisms that implement the Discretionary Access Control and Mandatory Access Control policies are contained with the access control function. For convenience, we represent the Mandatory Access Control and Discretionary Access Control mechanisms as a single implementation; we recognize that some products have separated the implementations into distinct modules. That separation is not significant to our discussion.

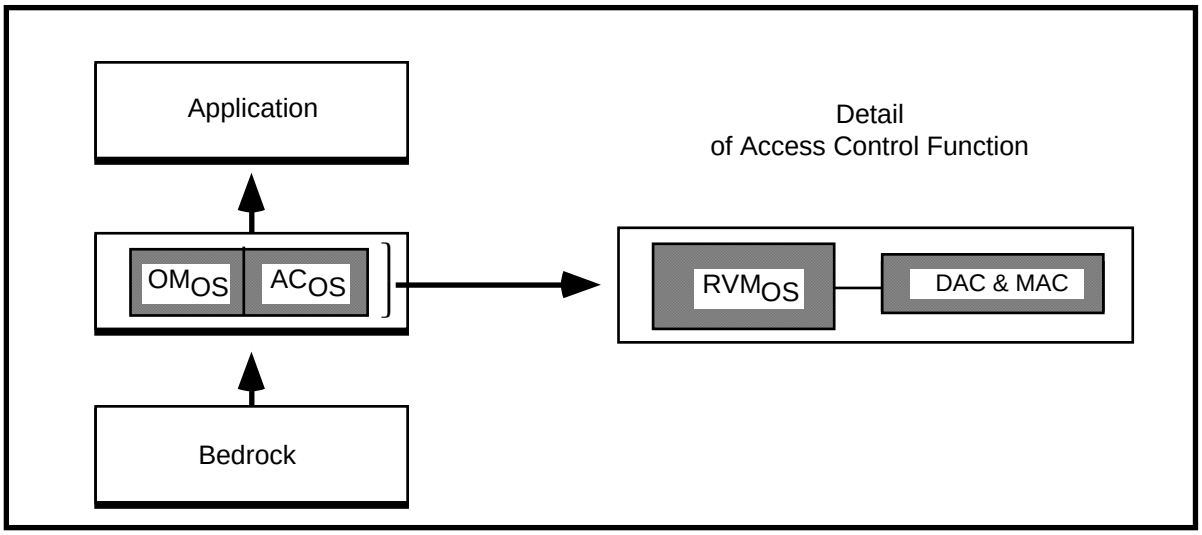

Figure 5. Conventional Discretionary Access Control and Mandatory Access Control Implementation

The traditional non-subsetted Trusted Computing Base, while serving as useful framework for initial trusted products offerings, has a variety of limitations. One major limitation is that the concept is not adequate for evaluating the security properties of contemporary information technology systems. The monolithic approach collects Object Managers, Reference Validation Mechanism, and other functions into a well-defined, static Trusted Computing Base. Contemporary information technology systems view the operating system as a necessary building block, but emphasize integration of sophisticated applications, such as a database management system or e-mail system. Often these applications are more complex than the operating system. However, introducing additional applications, which often have a security-relevant function, is at odds with the concept of a well-defined, static TCB.

3.3 TRUSTED COMPUTING BASE SUBSETS APPROACH

Trusted Computing Base Subsets approach, as introduced in (Shockley, 1987) and developed in the *Trusted Database Interpretation* (NCSC, 1991) addresses the addition of a trusted application (such as a database management system) on a trusted operating system. The approach is a very attractive extension to traditional Trusted Computing Base concepts and has been used for other purposes (Sterne, 1991). The Trusted Computing Base Subset approach is based on the "divide and conquer" strategy. It views a "system Trusted

Computing Base" (M)[1] as composed of a collection of layers of Trusted Computing Base Subsets (M_i). The Trusted Computing Base Subset approach may be characterized as a vertical extension of the Trusted Computing Base concept.

The Trusted Computing Base Subset approach imposes a partially ordered relationship among the Trusted Computing Base Subsets. The user interacts with the uppermost layer (top Trusted Computing Base Subset). Each Trusted Computing Base Subset provides an additional Access Control Policy on an object obtained from a lower Trusted Computing Base Subset. From an Access Control Policy enforcement point of view, the Trusted Computing Base Subset approach assumes that the system Access Control Policy[2] is the logical AND of all the constituent Access Control Policies that are enforced by the Trusted Computing Base Subsets.

The *Trusted Database Interpretation* specifies that Trusted Computing Base Subsets must be arranged in a partial order to prevent circular chains of dependencies (NCSC, 1991, page 16). We note that this structuring of Trusted Computing Base Subsets makes it easy to concatenate Trusted Computing Bases, thereby forming the AND of the Access Control Policies enforced by the Trusted Computing Base Subsets. The discussion accompanying the development of the *Trusted Database Interpretation* concluded that it would be very difficult, if not impossible, to implement any combination of policies other than the logical AND employing Trusted Computing Base Subsets.

Various architectures have been proposed for the configuration of a trusted database management system running as an application on a trusted operating system. Although there are many variations, we represent a generic architecture depicting a database management application in Figure 6. The database management system obtains resources from the operating system and creates new objects (e.g., relations, views, elements).

There is no required relationship between the operating system access control policy and the database management system access control policy. In the SeaViews architecture (Denning, 1988), for example, the operating system provides the Mandatory Access Control policy component and the database management system provides the Discretionary Access Control access control policy component. Even if a category of policy is implemented by both the operating system and the application, the policies may differ in some details. For example, the Mandatory Access Control implemented in an operating system may permit write-up while the Database Management System Mandatory Access Control does not. Moreover, the Database Management System Access Control Policy may include policies in addition to

---

1 This notation was introduced in the *Trusted Database Interpretation* (NCSC, 1991).

2 "P" in the *Trusted Database Interpretation*.

those found in the Orange Book. Integrity policies, such as referential integrity, are relevant examples.

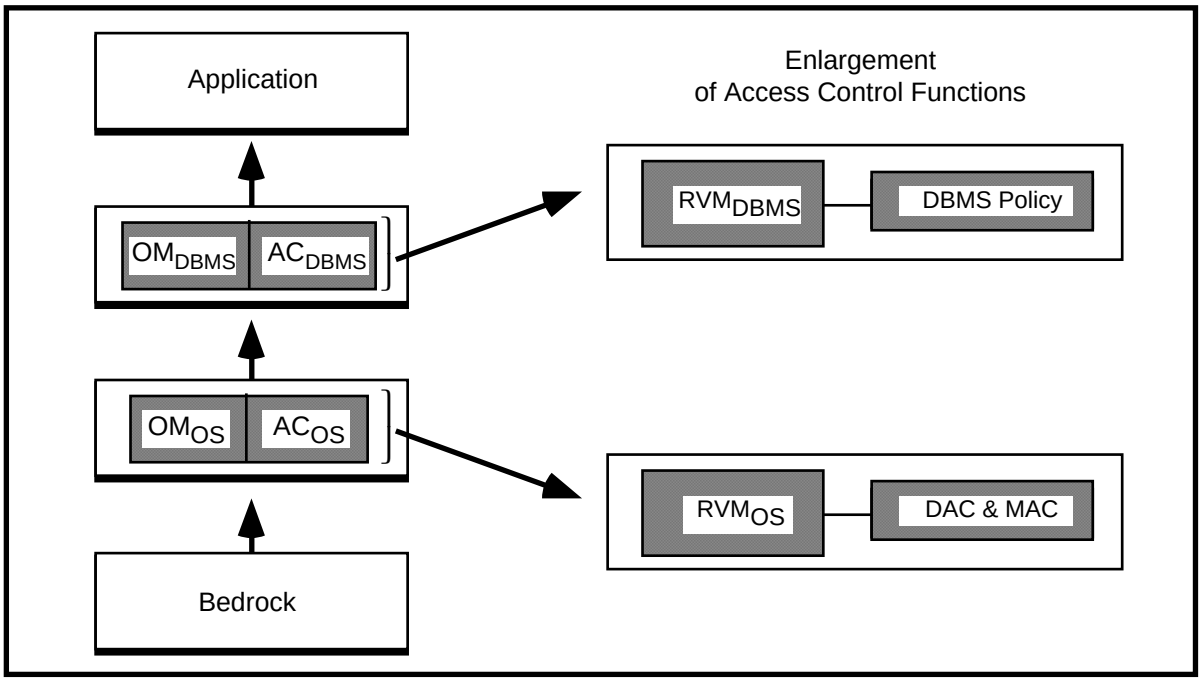

Figure 6. Trusted Database Management System Architecture

3.3.1 Relative Trust Among Subsets

The Trusted Computing Base is like a Jordan curve,[3] partitioning the hardware and software into two parts: that part which is within the Trusted Computing Base and that part which is outside the Trusted Computing Base. The part inside the Trusted Computing Base is referred to as *trusted* and the part outside the Trusted Computing Base is referred to as *untrusted*. This simple separation into trusted and untrusted categories suffices when there is a non-subsetted Trusted Computing Base. An extension is needed, however, to accommodate

[3] A non-self-intersecting, continuous, closed curve is called a *Jordan curve*. See any good text on complex analysis, such as Nehari, Z., 1961, *Introduction to Complex Analysis*, Allyn and Bacon, Boston.

multiple Trusted Computing Base Subsets.

In a environment in which there are multiple Trusted Computing Base Subsets, the concept of "trusted" needs to be generalized to "trusted with respect to." That is, one Trusted Computing Base Subset is trusted with respect to another Trusted Computing Base Subset. This concept was introduced in the *Trusted Database Interpretation* using the terminology "Trusted Computing Base Subset 2 is less primitive than Trusted Computing Base Subset 1." The relationship among three Trusted Computing Base Subsets can be expressed using the operator , "trusted with respect to," as:

$$\text{Trusted Computing Base}_1 \quad \text{Trusted Computing Base}_2 \quad \text{Trusted Computing Base}_3$$

While the *Trusted Database Interpretation* emphasizes the "trusted with respect to" relationship, we observe that the reverse of this relationship also holds. That is to say, one Trusted Computing Base Subset also exhibits the quality that one Trusted Computing Base Subset is "untrusted with respect to" another Trusted Computing Base Subset.

The hierarchy in Figure 7 illustrates these relationships. The hardware and software trusted with respect to Trusted Computing Base Subset 1 includes Trusted Computing Base Subset 1 itself and the bedrock. Trusted Computing Base Subset 2 and the application are untrusted with respect to Trusted Computing Base Subset 1. The hardware and software trusted with respect to Trusted Computing Base Subset 2 includes Trusted Computing Base Subset 2 itself, Trusted Computing Base Subset 1, and the bedrock. The application is untrusted with respect to Trusted Computing Base Subset 2.

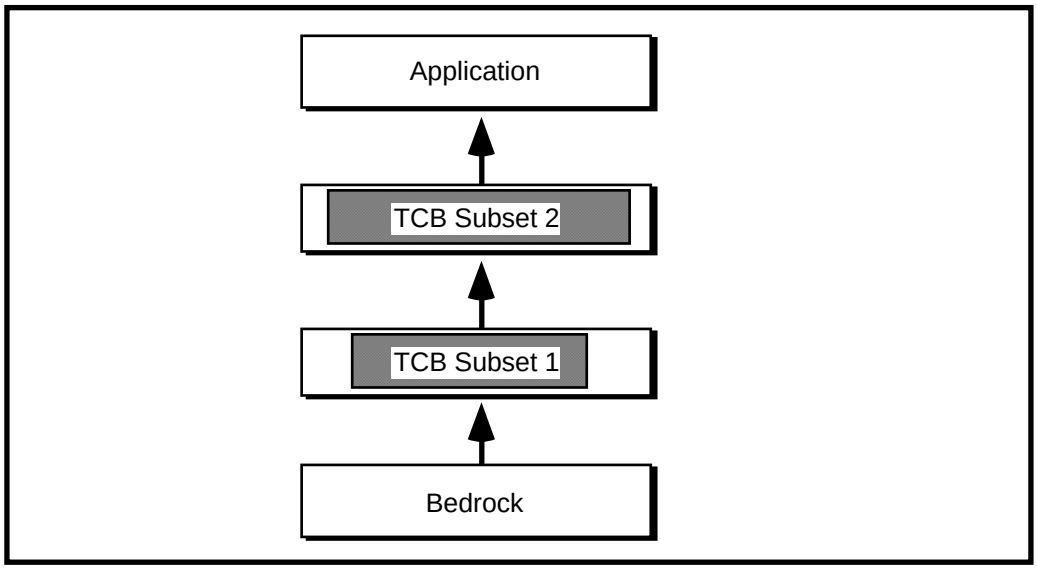

Figure 7. Trusted Computing Base Subsets Environment

3.3.2 Relating Trusted Computing Base Subsets to the Anderson Report

Any approach, including the Trusted Computing Base Subsets approach, that attempts to achieve the controlled sharing goal must satisfy the three guiding principles of Reference Validation Mechanisms. The principle that the Reference Validation Mechanism must be tamperproof is recognized in the *Trusted Database Interpretation* with reference to the need for extension of the domain concept: "Each candidate Trusted Computing Base Subset must occupy a distinct subset-domain such that modify-access to a Trusted Computing Base Subset's subset-domain is permitted only to that Trusted Computing Base Subset and (possibly) to more primitive Trusted Computing Base Subsets." An example using Multics rings is footnoted in the Anderson Report.

Note that subset-domains must be provided by some unspecified mechanism. A hardware basis is inferred, possibly with software extension as in virtual rings (Shockley, 1988) or type enforcement (Boebert, 1988). Another possibility is to rely solely on the hardware protection of Trusted Computing $Base_1$, with each succeeding, less privileged Trusted Computing Base Subset being protected by its more privileged antecedent. That is, for Trusted Computing $Base_i$ Trusted Computing $Base_j$, Trusted Computing $Base_j$ may be considered an object created and protected by Trusted Computing $Base_i$. Trusted Computing $Base_i$ controls access to Trusted Computing $Base_j$, thereby ensuring its protection from tampering. This approach,

however, imposes a rigid hierarchy on the ordering of the Trusted Computing Bases.

The principle that the Reference Validation Mechanism must always be invoked is no different when multiple Trusted Computing Base Subsets exist. When an application submits an access control request, that request must be validated by the Reference Validation Mechanism and, if access is granted, it must be consistent with the access control policy.

Concerning the requirement that the Reference Validation Mechanism must be small enough to be subjected to analysis and tests, we believe that the major advantage of the Trusted Computing Base Subset approach is that it adds structure to the Trusted Computing Base. Monolithic Trusted Computing Bases collect Object Managers, Reference Validation Mechanism, and other functions. The boundaries between these components is not always clear and can result in a large, baroque software structure.

The Trusted Computing Base Subset approach does not decrease the total size of the Trusted Computing Base, as the same components are still present. However, it does establish clear, well-defined lines between major components within the Trusted Computing Base, namely the Trusted Computing Base Subsets.

3.3.3 Additional Observations

The interaction of the Trusted Computing Base Subset constituent Access Control Policies is transparent to the user. The user only sees the logical AND of the Access Control Policies of the Trusted Computing Base Subsets.

Under the *Trusted Database Interpretation* approach, the addition of more Access Control Policies results in a more restrictive set of access controls on objects. A Trusted Computing Base Subset cannot reduce or eliminate access controls; it can only add restrictions. However, every Access Control Policy does not necessarily apply to every object. For example, Clark-Wilson (Clark, 1987) integrity explicitly does not control access to Unconstrained Data Items, and the Organization Controlled (ORGCON) policy (Abrams, 1991) subsumes Discretionary Access Control. New approaches to access control are required to address these situations

3.4 PARTITIONED TRUSTED COMPUTING BASE

The *Trusted Network Interpretation* introduced the concept of the horizontally Partitioned Trusted Computing Base, which it generally referred to as the *Network Trusted Computing Base*.[4] Partitioning may occur for functional or geographic reasons. Network Trusted

[4] This discussion is adapted from the *Trusted Network Interpretation.*

Computing Base Partitions may provide centralized network services, such as single (unitary) login or audit data storage and analysis, or may provide all Trusted Computing Base services to a subsystem that is part of a large distributed system.

Like a stand-alone system, the network as a whole possesses a single Trusted Computing Base consisting of the totality of security-relevant portions of the network. But, unlike a stand-alone system, the design and evaluation of the network rest on an understanding of how the security mechanisms are distributed and allocated to various components, in such a way that the security policy is supported reliably in spite of (1) the vulnerability of the communications paths and (2) the concurrent, asynchronous operation of the network subsystems.

A Network Trusted Computing Base that is distributed over a number of network subsystems is referred to as *partitioned*; that part of the Network Trusted Computing Base residing in a given subsystem is referred to as a *Network Trusted Computing Base partition*. A network host, also known as an *end system*, may possess a Trusted Computing Base that has previously been evaluated as a stand-alone system. Such a Trusted Computing Base does not necessarily coincide with the Network Trusted Computing Base partition in the host, in the sense of having the same security perimeter. Whether it does or not depends on whether the security policy for which the host Trusted Computing Base was evaluated coincides with the network security policy, to the extent that it is allocated to that host.

Some subsystems, such as the hosts, may satisfy the entire security policy in isolation; others, such as packet switches and access control centers, may have more specialized functions that satisfy only a subset of the network security policy. In addition, distinct subsets of the network security policy may be allocated to different network subsystems.

The allocation of a requirement to a subsystem does not simply mean that the subsystem satisfies the requirement in isolation, but includes the possibility that it depends on other subsystems to satisfy the requirement locally, or cooperates with other subsystems to ensure that the requirement is satisfied elsewhere in the network.

Figure 8 illustrates a contemporary system architecture and the partitioning of a Network Trusted Computing Base. Two server subsystems are shown, each implementing a Network Trusted Computing Base function. Trusted protocols would be employed for communication among the Network Trusted Computing Base partitions in all of the subsystems. The security mechanisms incorporated in these protocols would reflect the security policy implemented by the Network Trusted Computing Base.

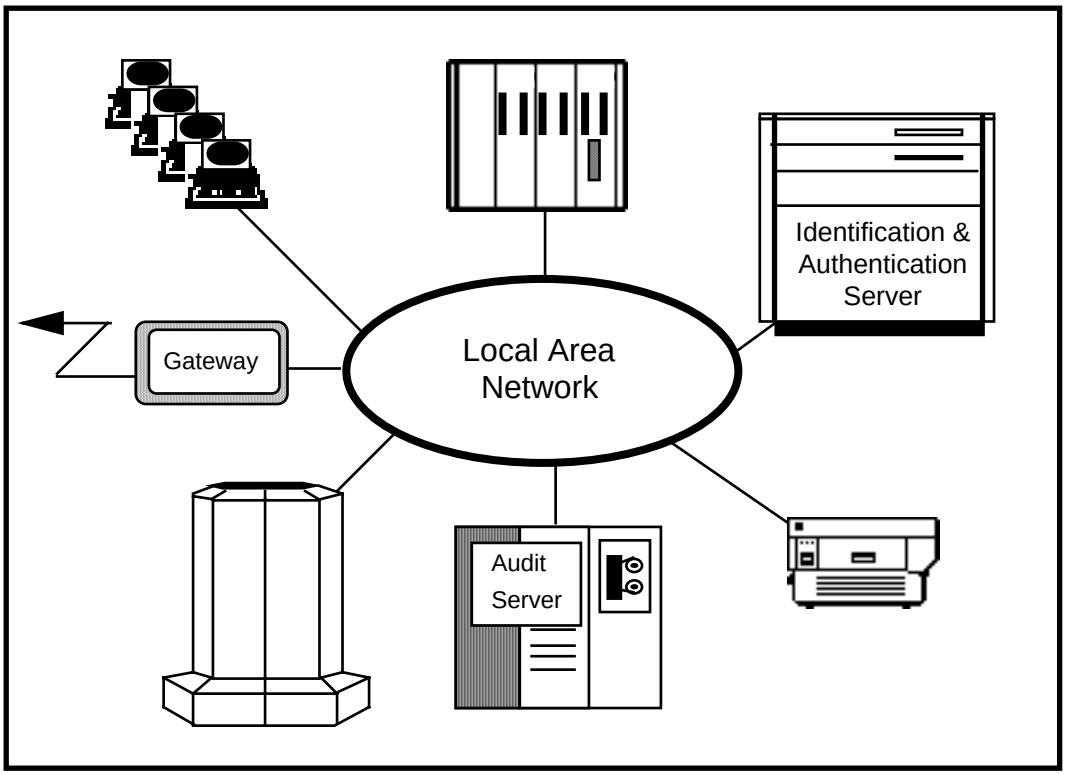

Figure 8. Network Trusted Computing Base

4 OTHER TOPICS IN ACCESS CONTROL

Trusted systems concepts have continued evolve and broaden. As confidence is gained with trusted technology and new applications considered, there are demands for new capabilities. One area of investigation involves new or alternative approaches to access control. These investigations are wide-ranging and include non-access control policies, the role of separation kernels to deal with complexity, and the presence of multiple access control policies. This section examines several of the more significant trends in trusted concepts and the needs that they address.

4.1 POLICIES OTHER THAN ACCESS CONTROL

The audit server in Figure 8 illustrates a non-access control policy. Identification and authentication supports audit and access control, but is not, itself, directly part of access

control. While the focus of this paper is on access control, we would be remiss in not pointing out that other policies may be part of the Trusted Computing Base's responsibility. Any process that can modify an attribute employed in enforcing a security policy must be part of the Trusted Computing Base.

An especially interesting case occurs when the security level of an output is lower than that of an input. Cryptographic methods are frequently used to produce objects of lower sensitivity so that less administrative and physical security measures are required for their protection. Communication and storage object are encrypted. The cryptography must be recognized by the Trusted Computing Base to permit the write down. Conventionally, the cryptographic processes are exempted from mediation by the Trusted Computing Base—they are considered trusted.

Computer security and communications security technologies are merging into Information Technology security. Software and firmware are replacing hardware in the implementation of cryptography. Assurance and evaluation methods are perforce merging. As discussed in (Brierley, 1992), the Canadian Trusted Computer Product Evaluation Criteria contains provisions to address part of this merger.

Another example of write down occurs when the information content is filtered or sanitized to reduce the sensitivity. Automated security guards are a very attractive way to achieve trusted transfers across security boundaries. While providing less integration than multilevel security in a single system, guards permit interconnection of systems operating at different, often system-high, security levels. Communication employing fixed-format messages with reliable labels is quite amenable to automated filtering and downgrading. Free-form text sanitation is much more of a research problem. (Neugent, 1989) contains an authoritative overview.

4.2 SEPARATION KERNELS

Traditional approaches to trusted Information Technology products and systems treat the Trusted Computing Base as residing in one protection domain and all other functions as residing in another domain. Both analytic and practical considerations favor the introduction of additional domains.[5] As more policies are added, the Trusted Computing Base could become too large and unwieldy for high-assurance modeling and security analysis. Such

[5] The definition of *domain* is somewhat stretched. *Domain* is defined in the Orange Book as "the set of objects that a subject has the ability to access." As multiple policies and authorities are introduced, the term *protection domain*, meaning of the bounds on authority and policy, is introduced.

complexity becomes inevitable when multiple policies, especially those enforced by applications, are present.

Rushby (1984) views separation kernels, or domain separation, as being the appropriate base on which to build trusted systems. The goal of separation is rigorous isolation to gain such properties as integrity, confidentiality, or Trusted Computing Base self-protection. Appropriate separation prevents the accidental intermingling of information protected by different Access Control Policies, prevents the inappropriate transfer of information, and provides the structural integrity necessary to support appropriate Access Control Policies.

In considering Trusted Computing Bases for embedded systems, Rushby proposed that a trusted embedded computer system should be structured in three layers. The lowest layer was a domain separation mechanism, the middle layer contained a set of Resource Managers, and the highest layer was the set of applications. The domain separation provided by the domain separation mechanism allowed untrusted programs to operate in their protection domains and not interfere with one another. Both the domain separation mechanism and the Resource Managers contained instances of the Reference Validation Mechanism for adjudicating interdomain communications and access to resources, respectively.

This scheme offers an important advantage to the Trusted Computing Base Subset approach. Significantly, there is no dependency relationship among the protection domains. The "trusted with respect to" concept and operator introduced earlier for hierarchical Trusted Computing Base Subsets are not required in this environment. Since the protection domains are independent of each other, it is not necessary to impose an ordering relationship on the use of objects as resources as was done in the *Trusted Database Interpretation*.

4.3 EXTENDED ACCESS CONTROL FRAMEWORK

The Access Control Framework (ISO, 1993) provides a solid foundation for viewing the access control process. It identifies several component functions within the access control decision-making process. One function, the Access Control Decision Function, decides whether an initiator (i.e., the subject) may perform an action on a target (i.e., the object). The result of its decision-making process is a decision. The second function, the Access Control Enforcement Function, receives the decision and enforces it.

Not only does the Access Control Framework identify specific functions, it also identifies principal components used in the implementation of access control: Access Control Rules, Access Control Information, and Access Control Authority. The Access Control Rules, a derivative of natural language policies, establish the constraints on actions performed by the initiator. The Access Control Information is that information made available to an Access Control Decision Function about the initiator, target, and action, including contextual

information, that the Access Control Decision Function needs to make its access control decision. The Access Control Authority is the source of Access Control Information and Access Control Rules.

Abrams (1993) proposes an extension to the ISO Access Control Framework to addresses the presence of multiple access control policies. The current ISO Access Control Framework specifies a single set of Access Control Rules and a single Access Control Policy. In Abrams' proposed extension to the ISO Access Control Framework, a Metapolicy Function is introduced which combines the votes of multiple Access Control Decision Functions. The Metapolicy Function, logically positioned between the Access Control Enforcement Function and the Access Control Decision Functions, combines the votes of the various Access Control Decision Functions into a single vote passed to the Access Control Enforcement Function. The way in which these votes are combined is defined by a set of Metapolicy Rules.

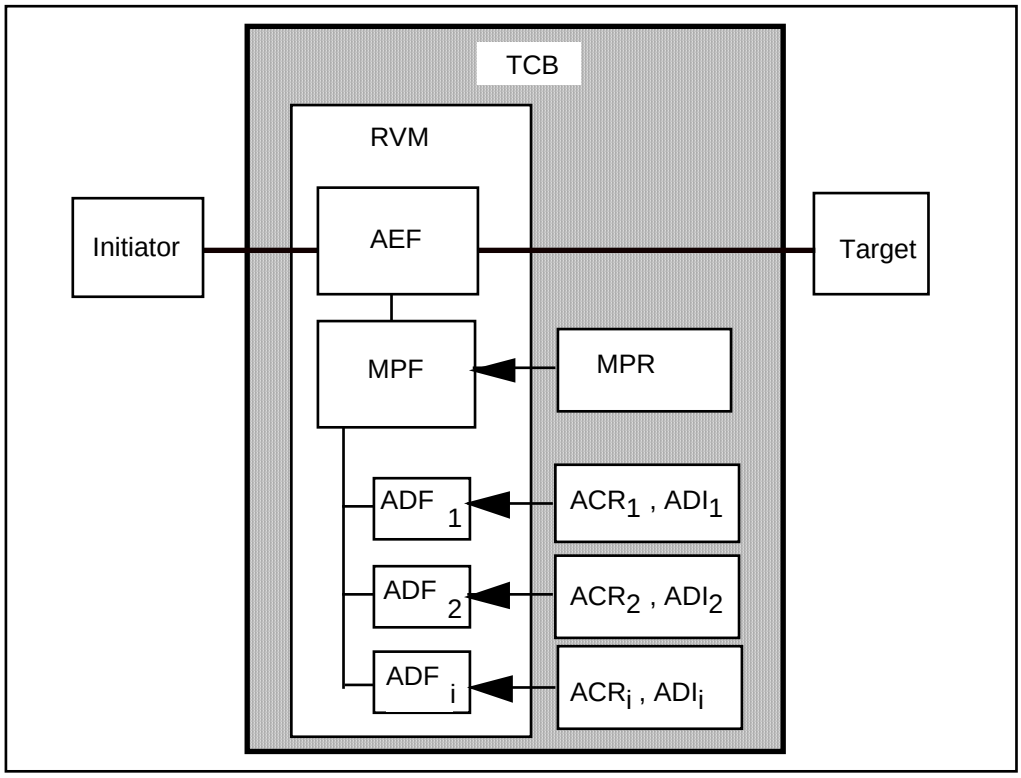

Figure 9. Component Placement in the ISO Access Control Framework

The functions of the Access Control Framework can be related to the functions identified in the Anderson Report. The Access Control Enforcement Function, the Metapolicy Function, and the Access Control Decision Functions are the access enforcement components. These components, working together to mediate each access by an initiator to a target, constitute the Reference Validation Mechanism. The Metapolicy Rule, on the other hand, contains the specific rules to be implemented by the Metapolicy Function. These rules are the basis on which the Metapolicy Function combines the votes and makes the final access control decision. These rules do not have to be part of the Reference Validation Mechanism. However, to ensure the correct, consistent operation of the Metapolicy Function, the Metapolicy Rule must be located within the Trusted Computing Base to ensure that it is adequately protected. For similar reasons, the Access Control Rules do not have to be part of the Reference Validation Mechanism but must be located within the Trusted Computing Base (see Figure 9).

5 SUMMARY

This second paper in the series has examined the evolution in the understanding of trusted information technology systems and the structure of security functions. The discussion began with an examination of the relevance of object management in achieving security goals. With this orientation in place, the structure of a conventional trusted system was examined. This discussion developed a parallel between the object management and access control functions in the operating system and Policy-Enforcing Applications. Significantly, with the introduction of Policy-Enforcing Applications, trust is required in additional components, not just the operating system. The result is an increase in the size and complexity of the TCB.

Trusted systems concepts continued evolve and broaden. As confidence is gained with trusted technology, new applications are considered. This, in turn, creates demands for new capabilities. This installment examined new or alternative approaches to access control including non-access control policies, the role of separation kernels to deal with complexity, and the presence of multiple access control policies. The next installment in this series continues the examination of new trends in trusted technology.

ACKNOWLEDGMENTS

The authors wish to acknowledge the ideas and encouragement received from the anonymous reviewers and colleagues who read prior drafts of this paper: Ed Amoroso, Blaine Burnham,

David Gomberg, Jody Heaney, Ralf Houser, Dale Johnson, Len LaPadula, Carol Oakes, and Jim Williams. This work was supported by the National Security Agency under contract DAAB07-93-C-N651 and by the MITRE Corporation.

LIST OF REFERENCES

Abrams, M. D., J. Heaney, O. King, L. J. LaPadula, M. Lazear, and I. M. Olson, October 1991, "Generalized Framework for Access Control: Towards Prototyping the Organization Controlled Policy," *Proceedings of the 14th National Computer Security Conference.*

Abrams, M. D., and M. V. Joyce, 1993, "Extending the ISO Access Control Framework for Multiple Policies," *Proceedings of the Ninth International Information Security Conference,* Elsevier Science Publishers.

Anderson, J. P., October 1972, *Computer Security Technology Planning Study*, ESD-TR-73-51, Vol. I, AD-758 206, ESD/AFSC, Hanscom AFB, Bedford, MA.

Boebert, W. E., 1988, "The LOCK Demonstration," *Proceedings of the 11th National Computer Security Conference*, Baltimore, MD.

Brierley, W., 1992, "Integrating Cryptography into Trusted Systems: A Criteria Approach," *Proceedings of the Eighth Annual Computer Security Applications Conference,* San Antonio, Texas.

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

Denning, Dorothy E., et al., April 1988, "The SeaViews Security Model," *Proceedings of the 1988 IEEE Symposium on Research in Security and Privacy*, Oakland, CA, IEEE Computer Society Press, pp. 218-233.

International Organization for Standardization (ISO), September 1993, International Electrotechnical Committee, Joint Technical Committee 1, Subcommittee 21, *Information Technology - Open Systems Interconnection - Security Frameworks in Open Systems - Part 3: Access Control*, Document Number ISO/IEC JTC/SC 21 N 8224 (or most recent draft).

National Computer Security Center (NCSC), April 1991, *Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria*, NCSC-TG-021,

Version 1.

Neugent, W., 1989, "Guidelines for Specifying Security Guards," *Proceedings of the 12th National Computer Security Conference*, pp. 320-338.

Rushby, J., September 1984, "A Trusted Computing Base for Embedded Systems," Proceedings *of the 7th Department of Defense/NBS Computer Security Conference*, pp. 294-311.

Shockley, W. R., and R. R. Schell, December 1987, "Trusted Computing Base Subsets for Incremental Evaluation," *Proceedings of the Third Aerospace Computer Security Conference*, American Institute of Aeronautics and Astronautics, CP 8711.

Shockley, W. R., T. F. Tao, and M. F. Thompson, 1988, "An Overview of the GEMSOS Class A1 Technology and Application Experience," *Proceedings of the 11th National Computer Security Conference*, pp. 238-245.

Sterne, D. F., November 1991, *A Trusted Computing Base Subset for Integrity and Role-Based Access Control, Trusted Information Systems*, Report R-388.