Trusted System Concepts

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 first of three related papers exploring how contemporary computer architecture affects security. Key issues in this changing environment, such as distributed systems and the need to support multiple access control policies, necessitates a generalization of the Trusted Computing Base paradigm. This paper develops a conceptual framework with which to address the implications of the growing reliance on Policy-Enforcing Applications in distributed environments.

1 INTRODUCTION

A significant evolution in computer software architecture has taken place over the last quarter century. The centralized time-sharing systems of the 1970s and early 1980s are rapidly being superseded by the distributed architectures of the 1990s. As an integral part of the architecture evolution, the composition of the system access control policy has changed. Instead of a single policy, the system access control policy is more likely to be a composite of several constituent policies implemented in applications that create objects and enforce their own unique access control policies.

This paper first provides a survey that explains how the security community developed the accepted concepts and criteria that addressed the time-shared architectures. Second, the paper focuses on the changes currently ongoing, providing insight into the driving forces and probable directions. This paper presents contemporary thinking; it summarizes and generalizes vertical and horizontal extensions to the Trusted Computing Base (TCB) concept. While attempting to be logical and rigorous, formalism is avoided.

This paper was first published in Computers & Security, Vol. 14 No.1 pp. 45-56, © Elsevier Advanced Technology 1995, Oxford, UK; http://www.elsevier.nl/locate/compsec. This is the first of three related papers.

1.1 ORGANIZATION OF THE PAPER

The evolution of current systems reliant on applications, not just the operating system, for trusted services leads into a discussion of trusted computing systems concepts. Section 2 establishes a framework for understanding of the goals and purpose of trusted information technology. Section 3 discusses how security attributes play a key role in security policy enforcement. Section 4 presents access as a specific type of interaction between a subject and an object that results in the flow of information from one to the other and an access control policy to mediate that flow. Section 5 presents separation as an access control mechanism having the goal of separation is rigorous isolation to gain such properties as integrity, confidentiality, or TCB self-protection. Section 6 introduces a concept, called bedrock , to deal with the layered dependencies of hardware and software. Risk management addresses the trustworthiness of the bedrock layers. Section 7 addresses the problems in multidomain security, arising from domains enforcing different security policy possibly under the control of different security authorities. Section 8 concludes the paper with a discussion of assurance in terms of effectiveness and correctness and the difficulties in establishing correctness.

1.2 EVOLUTION OF COMPUTER TECHNOLOGY

This section discusses our view of the evolution of how systems are built. The key factor is that current systems are reliant on applications, not just the operating system, for essential services. In trusted systems, these essential services include security.

Early computers were composed of a single central processing unit with a very limited addressing capability and a primitive operating system that provided file management and peripheral equipment services. With the introduction of time sharing, local users could interact with the computer using terminals directly connected to the mainframe; as the communications systems evolved, remote users could similarly access the mainframe computing resource. Contemporary systems are more likely to be composed of a collection of cooperating components located on workstations interconnected across a high-speed local area network or wide area network, such as the Internet. Early and contemporary system models are shown in Figure 1.

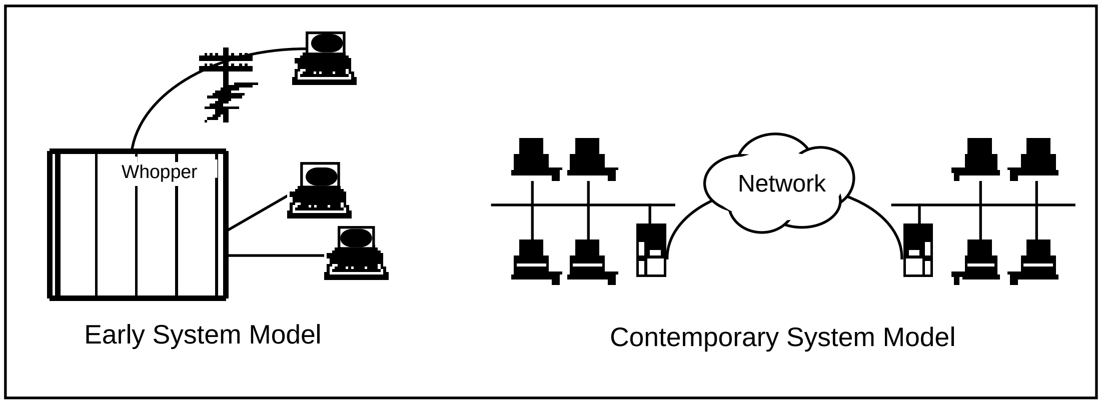

Figure 1. Early and Contemporary System Models

Early applications were built from programs that were narrow in focus and limited in function, often written by an internal development organization in assembly language. In an operational context, these applications more often resembled stovepipes having little integration with other applications. Current application systems are likely to be composed of several commercial off-the-shelf programs with a high degree of integration among the individual programs. The result is an application system that provides the end user with access to all of the corporate information assets and an array of processing capabilities for manipulating that information.

Figure 2 contrasts the software architecture of an early application and a contemporary application to illustrate the evolution in the composition of applications. Early application programs interacted with the operating system. The architectures of contemporary applications are much more complex. Contemporary applications draw upon services and resources from a number of other applications, not just the operating system. Applications such as the communications system to exchange files, middleware products to access data in a transparent manner, and an e-mail system to provide message services among users are essential building blocks for contemporary application systems.

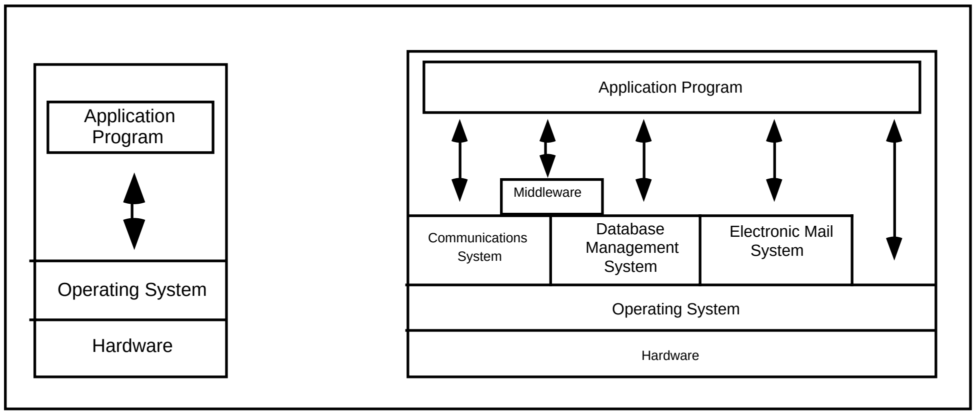

Figure 2. Building Blocks of an Application System

2 TRUSTED SYSTEMS CONCEPTS

The first section presented a framework for the current thinking about the organization of contemporary application systems. We now focus our attention on trusted computing systems concepts. This section begins the discussion by establishing an understand of the goals and purpose of trusted information technology.

The understanding of trusted technology and trusted computing concepts, tracking the evolution of products and needs in the commercial sector, has evolved since the introduction of the technology and concepts in the early 1970s. The Anderson Report (Anderson, 1972) formalized several important concepts that continue to be fundamental in information security, even after 20 years. The Reference Monitor concept was introduced as an ideal to achieve controlled sharing. “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*, for which the Anderson Report adds the following guiding principles:

1. The Reference Validation Mechanism must be tamperproof.

2. The Reference Validation Mechanism must always be invoked.

3. The Reference Validation Mechanism must be small enough to be subjected to analysis and tests to ensure that it is correct.

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 TCB. A TCB 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 TCB boundary. The reader is cautioned that other authors are not always as careful as they might be in using these terms. Another term, not used in this paper, is security kernel. Equivalent terminology, Security Enforcing and Security Relevant, has been introduced in the *Information Technology Security Evaluation Criteria* (ITSEC) (Commission, 1991). *Security Enforcing* refers to the hardware, firmware, software, and data which directly contributes to satisfying the security objectives. We understand security enforcing to be equivalent to Reference Validation Mechanism. *Security Relevant* refers the hardware, firmware, software, and data which is not security enforcing, but must function correctly or be correct in order that the security enforcing functions can enforce security. We anticipate further evolution of terminology; the reader should be careful to understand how any specific author uses terminology.

The third principle of the Reference Validation Mechanism needs a little refinement:

3a. The Reference Validation Mechanism must be comprehensible enough to be subjected to analysis and tests to ensure that it is correct.

Size, structure, and programming language are among the principal contributors to comprehensibility of a system. Contemporary applications are complex, often more complex than early operating systems. This complexity creates a tension between implementing a system that meets its functional goals and creating an implementation that is comprehensible. Software engineering techniques and high-level languages are elements that contribute to achieving comprehensibility. High-level languages, for example, provide constructs that increase the capability to express ideas and provide significant advantages in readability and writability. These qualities contribute to establishing the correct operation of the application. Besides contributing to meeting the security goals, high-level languages also enhance the reliability and maintainability of the application.

Building on the framework established by the Anderson Report, Bell and LaPadula (1975) developed a lattice-based formal model that was analogous to the manual protection of national security documents. The Bell-LaPadula model continues as the dominant security policy model even today. Biba (1977) developed an integrity policy model that is a dual of the Bell-LaPadula model in that it inverts the dominance relation. While Biba and Bell-LaPadula are isomorphic, they are sometimes treated as separate policy models. See (Sandhu, 1993b) for a complete up-to-date treatment.

With its arrival in the early 1980s, the U.S. Department of Defense *Trusted Computer System Evaluation Criteria* (TCSEC), also known as the Orange Book (DOD, 1985), codified many of the prevailing computer security concepts. The Orange Book provided a solution space for two sets of access control policies and mechanisms: Discretionary Access Control and Mandatory Access Control. The Orange Book defines discretionary access control as "a means of restricting access to objects based on the identity of the subjects and/or groups to which they belong. See (Downs, 1985) for a classic discussion of issues. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps) indirectly on to any other subject." Mandatory policy implies that the authorization is outside the control of a typical user (Saltzer, 1975). In the Orange Book, *mandatory* is the complement of *discretionary*.

The Orange Book definition of *mandatory security* policy and mechanisms is that mandatory security works by associating labels, one form of a security attribute, with objects to reflect their sensitivity. Similar labels are associated with subjects to reflect their authorizations. Under the mandatory policy, every subject runs in an execution domain implied by the label (Shirey, 1981). The Reference Validation Mechanism compares the labels associated with subjects and objects, and grants a subject access to an object only if the result of the comparison indicates that the access is proper according to the security policy. The security policy in the Orange Book is expressed in a *dominance* comparison that satisfies three well-known mathematical conditions: (1) reflexivity, (2) antisymmetry, and (3) transitivity. Dominance reflects a set of rules for comparing access classes. Some flows are allowed and others forbidden. This concept of information flow policy was formally defined by Denning (1976).

The Orange Book addressed those policies for which a consensus about computer implementation existed at the time; other policies were not addressed. For example, although Director of Central Intelligence Directive (DCID)1/7 (DCI, 1981) preceded the Orange Book, the policies it specifies are not (well) addressed by the Orange Book. The Originator Controlled policy in DCID 1/7 has motivated development of non-discretionary access controls[1] by Graubart (1989), McCollum (1990), Abrams (1991), and Sandhu (1992). These works are summarized in (Abrams, 1993b).

1 The earliest work in this area known to the authors was conducted by K. Rogers (then) of UNISYS in 1986. Unfortunately, no public reference is available.

Some circles appear to have rejected the terminology of the Orange Book, Reference Validation Mechanism, or TCB because of its origins. This is a case of throwing out the baby with the bath water. Analysis of many, but not all, of the security concerns of civil government and commercial organizations uncovers much more similarity than some people suppose. The assertion that the Orange Book completely covered the field did not help acceptance outside the community in which it was published.

Looking back at innovations in access control policies, the two policies defined in the Orange Book can be viewed as end points in a continuum of access control policies as defined by who has control over the policy. The *non-discretionary* policy identifies the situation in which authority is vested in some users (i.e., the system administrators), but there are controls on delegation and propagation of authority. The other end of the continuum, the *discretionary* access control policy, represents the case in which authority is extended to all users. This relationship can also be viewed as a tree (Abrams, 1993b) in which *discretionary* is the case in which the branches extend to every user, and *non-discretionary* is the case in which there are branches that do not extend to every user. Saltzer (1975) illustrates this point quite clearly with hierarchy of controllers and non-discretionary Access Control List (ACL) use.

Innovative approaches continue to appear, especially for dealing with policies other than the two confidentiality policies contained in the Orange Book. Moffett and Sloman (1988) address the source of policy in non-defense organizations. Moffett (1993) views policy as composed of two components: an imperatival policy that controls initiating or inhibiting actions from the authority policy that permits or prohibits the actions. In the framework, the functions that implement the policy components are separated into a Situation Monitor and Reference Monitor, respectively. In addition to the Biba integrity model, Clark and Wilson (1987) introduced a Commercial Integrity Model. See Abrams (1993c) for some insights on this model. Millen (1994) proposes a Denial-of-Service Protection Base implementing waiting-time policies and user agreements. We believe that the separation kernel design described in section 5 accommodates all such innovations.

The *Trusted Network Interpretation* (NCSC, 1987b) addresses a single network security policy that may require data secrecy, data confidentiality, or both. The *Trusted Database Interpretation* (NCSC, 1991a) provides interpretations that extend the Orange Book “to any software system which supports sharing and needs to enforce access controls.” The *Trusted Database Interpretation* uses Database Management Systems as representative of such systems; in this paper, we use the more general term *policy-enforcing applications*. It is a source of disappointment to the database security community that no official documents addressing the particular needs of database security have been published. The interested reader should see (Sandhu, 1993a) for identification of the database security issues.

3 SECURITY ATTRIBUTES

Security attributes play a key role in security policy enforcement. Security attributes are the characteristics or properties that are used by the rules that enforce security policy. A sample of the characteristics or properties used as security attributes include the user's name, group, role, employer, citizenship, clearance, classification, and location. The choice of security attributes depends upon the specific control policy in force.

There are many ways to organize these security attributes. Such design choices have profound impact on efficiency and ease of use. Using the Discretionary Access Control policy requirement cited in the Orange Book as an example, the security attributes supporting this policy can be associated with either subjects or objects (NCSC, 1987a).

Additional treatments of security attributes are emerging. ECMA Standard 138 (1989) addresses interchange of security attributes among security domains. The draft International Organization for Standardization (ISO) Access Control Framework (ISO, 1993) contains some excellent concepts; this work is evolving as the document makes its way toward international standardization. While this ISO standards work is, by charter, solely concerned with access control, its ideas generalize nicely to a broader concern with Information Technology security. Its treatment of access control attributes generalizes painlessly to security attributes.

The ISO framework introduces a very useful distinction that security attributes may be *associated with* a person, process, object, or context. Independently, the security attributes may *be about* (i.e., refer to) persons, processes, objects, or contexts. For example, an access control list is a security attribute data structure about personal identifications associated with objects. This is a very useful general concept.

The ISO framework includes the concept of associating security attributes with contextual information. In particular, the ability of an individual to take on a role or to assume a group membership may depend upon contextual information such as the location of the initiator, the time of access, or the particular communications path chosen. Rules and mechanisms such as Access Control Lists can refer to this contextual information during the decision-making process. Although perhaps implicit in prior formulations, making this association explicit is a valuable contribution.

The Logical Coprocessing Kernel (LOCK) and Clark-Wilson (Clark and Wilson, 1987) distinguish between the security attributes associated with users and processes. Associating security attributes with programs supports configuration management and provides protection against malicious modification of programs by Trojan horses and viruses. Other security attributes associated with programs can restrict the operations permitted by specified programs on specified objects when they are executed.

4 ACCESS CONTROL

In a computer system, access is a specific type of interaction between a subject and an object that results in the flow of information from one to the other. Exactly what the subject does when it accesses the object is further defined by introducing access modes such as read, read-write, write, and execute. Access control deals with the limits on the interactions between subjects and objects. These limits, expressed as an access control policy, are enforced by an access control mechanism that is designed to detect and prevent unauthorized access and to permit authorized access.

A wide range of access control mechanisms has been introduced to achieve access control goals in a computer system. An access control mechanism is the combination of hardware and software that adjudicates an access request and makes the decision either to permit or deny the access request. Saltzer identified the importance of descriptor based hardware protection systems as an access control mechanism. Special registers, called *descriptor register*, are interposed into the path to memory. In its most elementary form, the descriptor register is a simple base-bound mechanism that establishes which parts of memory are accessible to a process. As the concept evolved, more sophisticated versions of the descriptor mechanism added read, write, and execute access controls to the basic base-range control. This scheme was used on the Burroughs B5700/6700 and Multics systems and is available on some contemporary systems. A variation of this scheme, called *tag memory*, associates additional bits with every memory location. The additional bits control the type of access allowed.

Type enforcement is another access control mechanism that can be effectively implemented in software or hardware. Type enforcement is based on two sets of attributes. One set of attributes is associated with the current domain of execution of a subject. The other set of attributes is associated with an object. These attributes define the type of object. The access control mechanism restricts operations on an object based on the domain of execution of the subject attempting the access and the type of the object.

Work at Carnegie-Mellon University by Jones (1973) showed that type enforcement can be used for various aspects of (non-lattice based) memoryless subsystem problems (i.e., variants of the confinement problem). Jones and Wulf (1975) show that type enforcement can support non-discretionary access controls that can be represented as a lattice. Cohen and Jefferson (1975) illustrate that type enforcement can support a variety of non-discretionary policies that cannot be represented as a lattice. Boebert and Kain (1985) presented type enforcement as an alternative to Biba integrity. Type enforcement is not transitive, but it does allow information to flow from entity A to entity B through process C if the types of A, B, and C authorize that flow. LOCK (Boebert, 1988) employs type enforcement to provide pipelines. Pipelines can be used to implement functions that would be called trusted in a Bell-LaPadula architecture.

Clark and Wilson (1987) Transformation Procedures comprise a software-based access control mechanism. Clark and Wilson introduced Transformation Procedures to automate the concept of the well-formed transaction. The access privileges of a Transformation Procedure are determined external to the secure system and encoded in an access control triple of the form: (UserID, TPi, (CDIa, CDIb, CDIc, ...)), which relates a user, a Transformation Procedure, and the data objects that a Transformation Procedure may reference on behalf of that user. It appears that type enforcement and Transformation Procedures are quite similar; both constrain access of a process to a storage object based on security attributes. Further investigation into this similarity is indicated; it may well be that type enforcement and transformation procedure enforcement are the same concept by different names. The Transformation Procedure extends the TCB far beyond a Reference Validation Mechanism. Every process that affects integrity must be part of the TCB. There has been surprisingly little work done on integrity in recently years. To the authors' knowledge, no implementations of Clark-Wilson integrity have appeared. The very definition of integrity remains elusive. (NCSC, 1991b) provides an extensive overview of the diversity of the field, while (Abrams, 1993c) presents the views of a research study group.

5 SEPARATION

In the early 1980s, Rushby (1984) examined the use of a separation kernel as an access control mechanism. The goal of separation is rigorous isolation to gain such properties as integrity, confidentiality, or TCB self-protection. Separation on a single hardware platform emulates multiple hardware platforms. Separation has often been described as creating virtual computers, each virtual computer being a clone of the real hardware (Graff, 1992;Karger, 1990; Kelem, 1991). In actuality the virtual computers must share the real hardware; the task of the separation mechanism is to prevent any action on one virtual machine from affecting any other virtual machine. In other words, the separation mechanism prevents any information flow among virtual machines.

Hardware has become less expensive; virtual computers have been replaced with real computers as illustrated in Figure 1. Distributed systems don't emulate separate platforms; they are indeed composed of multiple computers. Separation can be provided as a architectural consequence in distributed systems or by a deliberate mechanism on a single CPU.

In considering TCBs 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 mechanism manages individual protection domains and the communications links between protection domains. 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.

Abrams and Joyce (1993) presented a new theoretical basis for composition of separately evaluated security modules. Principal components of this new framework included identification of Object Management, Access Control Decision, Access Control Enforcement, and Domain Separation as fundamental mechanisms on which the security of an automated information system depends. Expanding the notion of access control, Gate Keepers enforcing an interdomain communications policy were introduced to moderate information flow between protection domains. Complexity, policy and application selection, and distribution were identified as part of the problem space in which the systems architect works. Sterne (1994) proposes an alternative approach that builds on the TCSEC principles yet redraws the security perimeter of a trusted system so that it encompasses not only the TCB but also what they call the Controlled Applications Set, which we would call Policy Enforcing Applications. Their approach is based on layers, implementing TCB Subsets (Shockley, 1987; NCSC, 1991a).

6 BEDROCK CONCEPT

It is reasonable and logical that the mechanism(s) which implement an access control policy must be protected from (unauthorized) change. Our thinking is made easier and our confidence increases if there is a set of trusted resources, which we call *bedrock*, serves as the foundation of our security policies. This firm foundation is the basis of our trust. The interface to the bedrock specifies the set of resources used to build a trusted information technology system.

The bedrock concept is relative. The device designer, circuit designer, and operating system architect have different viewpoints. Each specialist assumes that the interface provided to him or her is trustworthy. This trust is a consequence of specialization. A person working at one technological level of abstraction is usually not prepared to investigate and determine the trustworthiness of the resources with which he/she works. For example, software experts rarely know about hardware design. However, they tend to trust the hardware. This trust may or may not be warranted. The hardware may be failure prone due to errors in design or fabrication; it may also have been built with malicious intent to sustain the same kinds of attacks as are commonly implemented in software, such as viruses and Trojan horses. See (Hampel, 1988) for further discussion.

Similarly, software experts who build TCBs or communications protocol interpreters are users of supporting software, such as compilers and editors. They assume that this supporting software is trustworthy. While this is usually the case, Thompson (1984) eloquently advises that one should be careful about extending trust. Recent work has described critical issues related to software trust and has proposed a set of criteria classes for measuring and comparing trust (Amoroso, 1991).

Addressing the trustworthiness of bedrock is a matter of risk management. Absolutely complete risk avoidance would address every possible level of risk. Risks might exist in the design of the chips, the side effects of instruction set design (especially unimplemented instructions in complex instruction set architectures), or the security flaws in all supporting software. It has been common when confidentiality was the only security policy to assume that mass produced bedrock was a sufficiently low risk that it could be ignored. Consideration of integrity and availability as security policies may justify reconsideration.

For the purposes of this paper, we assume an implementation in hardware or non-alterable firmware as the bedrock foundation. For each subsequent design, we assume that the Bedrock fundamental algorithms and mechanisms are trustworthy, including design, fabrication, and distribution. Our assumption specifically includes protection against modification. With this assumption, the bedrock and any TCB(s) built on the bedrock can be used as the building block for subsequent TCBs with transitivity of trustworthiness.

7 MULTIPLE ACCESS CONTROL POLICIES

With the passage of time, there has been an increased recognition that multiple security policies exist and should be automated. National defense classified and sensitive unclassified information, for example, are governed by different policies for non-disclosure. Even within the national defense classified communities, the executive branch departments determine their own policies. In the United States (U.S.), the Department of Energy and Department of Defense, for instance, have their own Access Control Policies. In the private sector, each enterprise is free to identify its own security policy. The consequence of this diversity is that there is no single pervasive access control policy, much less one implemented by a vendor of an information technology system.

A recent survey (Vazquez, 1994) presented the problems in multidomain security, which were defined as two systems, lying in two different security domains, trying to communicate securely. The two main reasons for problems are (1) the domains enforce different security policy and (2) the domains are under the control of different security authorities.

Options to address these needs are limited on current systems. The access control policy is an integral part of the operating system. Source code for an operating system is rarely available. Even if it is available, the technical skills to implement a change to the Access Control Policy may not be available. Moreover, any change to the operating system's TCB increases the cost of the certification and accreditation process.

Because of the conflict between organizational needs and the difficulty of introducing new Access Control Policies into the operating system, Access Control Policies are often implemented not in the operating system but in applications—in policy-enforcing applications. A company's specific security policy governing access to payroll information will be implemented in the company's payroll application, for example.

A contemporary statement of requirements for multiple security policy support, where security policy is much broader than just access control, may be found in the Department of Defense Goal Security Architecture (DOD, 1993). The DGSA specifies security principles and target security capabilities that guide system security architects in creating specific security architectures. Among the DGSA security principles are those that organize information in a manner this is consistent with mission objectives and that allow unique security policies to be applied to distinct collections of information. Thus, information systems that support multiple missions, must support multiple security policies simultaneously. Abrams (1993b) summarizes other prior work in the area of multiple access control policies, as does Hosmer's (1992a) survey of the evolution of thought concerning multiple policies. Part of the work on multiple policies included recognition of the need for a policy about policies, a metapolicy.[2] Work in this area is summarized in (Hosmer, 1992b).

8 ASSURANCE

Assurance is defined in the ITSEC as "the confidence that may be held in the security provided by a Target of Evaluation." Informally, assurance supports the belief that the system can be relied upon to reduce residual risk to the predetermined level. The ITSEC also observe that effectiveness and correctness both contribute to assurance. *Effectiveness* is determined by analysis of the specifications of the functional requirements; the environment in which the system will be used, the risks, threats, and vulnerabilities; and all the

[2] The earliest use of the term *metapolicy* known to the authors was by H. Hosmer (then) of The MITRE Corporation (Hosmer, 1990).

countermeasures, including physical, administrative, procedural, personnel, and technical. The system is considered effective if the result of this paper analysis is an acceptable residual risk. *Correctness* is determined by comparing the implementation of the countermeasures with their specification. The system is considered correct if the implementation is sufficiently close to the specification. This definition of correctness is compatible with the concept of risk management and is closer to the concept of *trustworthy* than to *error-free*. Contrasting effectiveness and correctness analysis we note that effectiveness is purely an analytic study of risks, threats, and countermeasures while correctness is concerned with the physical implementation of countermeasures in hardware, firmware, and software. In effectiveness analysis we decide that a certain combination of mechanisms is acceptable for use in a specified environment; in correctness analysis we determine if the implementation is sufficiently close to the specification.

In developing a TCB possibly involving a separation mechanism, you want assurance that it will enforce the stated security policies exactly by permitting the only the specified accesses and operations. Neither more nor less is acceptable. The TCB must do exactly what is identified in its specification and not do anything that is not so specified. Correctness always has to be with respect to a specification.

Abrams and Zelkowitz (1994) show that all known methods employed to establish correctness have shortcomings that make it impossible to establish correctness beyond reasonable doubt. That is, establishing correctness is a matter of belief, not proof. Selecting a method or combination of methods to establish correctness has been likened to a beauty contest. The major methods addressed by Abrams and Zelkowitz are mathematical models, simulation, testing, process models and procedures. Minor methods include structured programming, the spiral model, Computer Aided Software Engineering (CASE) tools, formal methods applied informally, object-oriented (OO) programming, reusing existing code, and process maturity. There is a growing consensus that no one technique can provide adequate assurance; see, for example, (Butler, 1993) or (Parnas, 1990). Despite years of experience, there is insufficient statistical or analytic evidence to support the selection of method(s).

The proceedings (NIST, 1994) of an Invitational Workshop on Information Technology (IT) Assurance and Trustworthiness identify crucial issues on assurance in IT systems to provide input into the development of policy guidance on determining the type and level of assurance appropriate in a given environment. Existing IT security policy guidance is based on computer and communications architectures of the early 1980s. Technological changes since that time mandate a review and revision of policy guidance on assurance and trustworthiness, especially since the changes encompass such technologies as distributed systems, local area networks, the worldwide Internet, policy-enforcing applications, and public key cryptography.

SUMMARY

In this paper we have reviewed trusted computer system concepts focusing on security attributes and access control extensions to the Reference Validation Mechanism. Bedrock and separation were introduced as architectural concepts, along with multiple access control policies, that characterize contemporary systems. The second paper in this series continues with a review of object management and extensions to the Trusted Computing Base concept.

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, January 1993, *On TCB Subsets and Trusted Object Management*, MTR-92W0000248, The MITRE Corporation.

Abrams, M. D., September 1993b, “Renewed Understanding of Access Control Policies,” *16th National Computer Security Conference.*

Abrams, M. D., E. G. Amoroso, L. J. LaPadula, T. F. Lunt, and J. G. Williams, November 1993c, “Report of an Integrity Research Study Group,” *Computers and Security*, Volume 12, No. 7.

Abrams, M. D. and M. V. Zelkowitz, October 1994, “Belief In Correctness,” *Proceedings of the 17th National Computer Security Conference*

Amoroso, Ed, et al., May 1991, “Toward an Approach to Measuring Software Trust,” *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy,* Oakland, CA, IEEE Computer Society Press, pp. 198-218.

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.

Bell, D. E., and L. J. LaPadula, June 1975, *Computer Security Model: Unified Exposition and Multics Interpretation,* MTR-2997, The MITRE Corporation, Bedford, MA.

Biba, K. J., April 1977, *Integrity Considerations for Secure Computer Systems*, ESD-TR-76-372, MTR-3153, The MITRE Corporation, Bedford, MA.

Boebert, W. E. and R. Y. Kain, 1985, “A Practical Alternative to Hierarchical Integrity Policies,” *Proceedings of the 8th National Computer Security Conference*, Gaithersburg, MD.

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

Butler, R. W., and G. B. Finelli, 12 January 1993, “The Infeasibility of Quantifying the Reliability of Life-Critical Real-Time Software,” *IEEE Transactions on Software Engineering,* Vol. 19, No. 1, pp 3-12.

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*.

Cohen, E., and D. Jefferson, November 1975, “Protection in the Hydra Operating System,” *Proceedings of the 5th Symposium on Operating Systems*, pp. 141-160.

Commission of the European Communities, 28 June 1991, *Information Technology Security Evaluation Criteria (ITSEC): Provisional Harmonized Criteria*, Luxembourg: Office for Official Publications of the European Communities, Version 1.2.

Denning, D. E., 1976, “A Lattice Model of Secure Information Flow,” *Communications of the ACM*, Vol. 19, No. 5, pp. 236-243.

Department of Defense, 1985, *Department of Defense Trusted Computer System Evaluation Criteria,* Department of Defense 5200.28-STD, Washington, DC.

Department of Defense, 1993, *Department of Defense Goal Security Architecture*, National Technical Information Service.

Director of Central Intelligence (DCI), 4 May 1981, *Control of Dissemination of Intelligence Information*, Directive No. 1/7.

Downs, D. D., J. R. Rub, K. C. Kung, and C. S. Jordan, 1985, "Issues in Discretionary Access Control," *Proceedings of the Symposium on Security and Privacy*, IEEE Computer Society, pp. 208-218.

European Computer Manufacturers Association (ECMA), December 14, 1989, Standard ECMA-138 — *Security in Open Systems: Data Elements and Service Definitions.*

Graubart, R., 1989, "On the Need for a Third Form of Access Control," *Proceedings of the 12th National Computer Security Conference*, pp. 296-303.

Graff, J., 1992, "Separation Machines," *Proceedings of the 15th National Computer Security Conference*, pp. 631-640.

Hampel, V. E., and C. F. Bender, 1988, "Covert Corruption of Integrated Circuits and Possible Strategies for Correction," *Proceedings of the First Conference on Hostile Intelligence Threat to Software, Firmware and Algorithms Embedded in U. S. Army Weapon Systems*, Defense Technical Information Center, Alexandria, VA.

Hosmer, H., 1990, "Integrating Security Policies," *Proceedings of the Third RADC Database Security Workshop, June 5-7, 1990, Castile, NY,* The MITRE Corporation, MTP 385.

Hosmer, H., October 1992a, "The Multipolicy Paradigm," *Proceedings of the 15th National Computer Security Conference.*

Hosmer, H., October 1992b, "Metapolicies II," *Proceedings of the 15th National Computer Security Conference.*

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).

Jones, A. K., 1973, *Protection in Programmed Systems*, Ph.D. dissertation. Carnegie Mellon University.

Jones, A. K., and W. A. Wulf, October-December 1975, “Towards the Design of Secure Systems,” *Software—Practice & Experience*, pp. 321-336.

Karger, P. A., et al., 1990, “A VMM Security Kernel for the VAX Architecture,” *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy,* Oakland, CA, IEEE Computer Society Press, pp. 2-19.

Kelem, N. L., and R. J. Feiertag, 1991, “A Separation Model for Virtual Machine Monitors,” *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy*, Oakland, CA, IEEE Computer Society Press, pp. 78-86.

McCollum, C. J., J. R. Messing, and L. Notargiacomo, 1990, “Beyond the Pale of MAC and DAC: Defining New Forms of Access Control,” *Proceedings of the Symposium on Research in Security and Privacy*, IEEE Computer Society Press.

Millen, J. K., 1994, “Denial of Service: A Perspective,” *Dependable Computing for Critical Applications*, Springer-Verlag (Proceedings of Conference in San Diego, CA, January 1994).

Moffett, J. and S. Sloman, February 1988, “The Source of Authority for Commercial Access Control,” *IEEE Computer*.

Moffett, J. D., Jonscher, D., and McDermid, J. A., 15 July 1993, *The Policy Obstacle Course: A Framework for Policies Embedded within Distributed Computer Systems,* Report SCHEMA/York/93/1. Dept. of Computer Science, University of York, England, .

National Computer Security Center (NCSC), 30 September 1987a, *A Guide to Understanding Discretionary Access Control in Trusted Systems*, NCSC-TG-003, Version 1.

______, 31 July 1987b, *Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria*, NCSC-TG-005, Version 1.

______, April 1991a, *Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria*, NCSC-TG-021, Version 1.

______, September 1991b, *Integrity in Automated Information Systems*, C Technical Report 79-91.

National Institute of Standards and Technology, 1994, *A Head Start: on Assurance, Proceedings of an Invitational Workshop on Information Technology Assurance and Trustworthiness,* NIST Internal Report 5472, Gaithersburg, MD.

Parnas, D. L., A. John van Schouwen, and Shu Po Kwan, June 1990, “Evaluation of Safety-Critical Software,” *Communications of the ACM*.

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.

Saltzer, J. H., and M. D. Schroeder, September 1975, “The Protection of Information in Computer Systems,” *Proceedings of the IEEE*, Vol. 63, No. 9, pp. 1278-1308.

Sandhu, R. S., and G. S. Suri, 1992, “Implementation Considerations for the Typed Access Matrix Model in a Distributed Environment,” *Proceedings of the 15th National Computer Security Conference*, pp. 221- 235

Sandhu, R. S. and Jajodia, S., 1993a, “Data and Database Security and Controls,” *Handbook of Information Security Management*, H. F. Tipton and Z. A. Ruthberg, editors, Auerbach Publishers.

Sandhu, R. S., November 1993b, “Lattice-Based Access Control Models,” *IEEE Computer*, pp. 9-19.

Shirey, L. J. and R. R. Schell, 1981, “Mechanism Sufficiency Validation by Assignment,” *Proceedings 1981 Symposium on Security and Privacy*.

Shockley, W. R. and R. R. Schell, “TCB Subsets for Incremental Evaluation,” *Proceedings Third Aerospace Computer Security Conference.*

Sterne, D. F., Glenn S. Benson, and H. Tajalli, June 1994, “Redrawing the Security Perimeter of A Trusted System,” *Proceedings 1994 Workshop on Fundamentals of Computer Security*.

Thompson, K., August 1984, “Reflections on Trusting Trust,” *Communications of the ACM*, Vol. 27, No. 8, pp. 761-763.

Vázquez-Gómez, J., April 1994, "Multidomain Security," *Computers & Security*.