Security Architecture-Based System Design

Edward A. Schneider
Institute for Defense Analyses
1801 N. Beauregard St.
Alexandria, VA 22311-1772
eschneider@ida.org

Abstract

We present a new view of information security based on concepts from the Defense Goal Security Architecture. This view looks at security according to the desire to protect and to share information without regard to either the hardware or the software architecture. The result is a separation of concerns and a security architecture that is based on system security requirements without including the network topology or the process interactions. The primary construct of the architecture is the information domain in which authorized users share information that has a common set of protection requirements. The system design is formed as a product of the security, the hardware, and the software architectures.

Keywords

Security architecture, security policy, information domain, information system design

1. Introduction

Traditional security models closely tie security to either the system or the software architecture. These models express security requirements in terms of services that either network gateways, operating systems and middleware, or software servers supply. The Defense Goal Security Architecture (DGSA) is an architectural framework in which system architects instead define security according to the requirements to protect information [2,3,6,8]. This definition is independent of which high-level software components operate on the information, or which network elements contain the information. The product of the security, software, and system architectures then forms a composite architecture that satisfies the constraints of each of its constituents. Separating the security requirements from those of computation and communication seems to greatly simplify these requirements.

1999 New Security Paradigm Workshop 9/99 Ontario, Canada

This paper presents the use of a security architectural framework in terms of the concepts of the DGSA. Rather than simply promoting these concepts, we demonstrate the benefits of a system design that starts with a definition of security that is separate from the system and the software architectures. As a result, our presentation of the DGSA concepts is somewhat different from that in [2] which intermingles the description of the security architecture with its interaction with the other architectures.

We start with a description of architectures. We then present the security architecture. This is followed by a discussion of how this architecture interacts with the system and the software architectures. We conclude with some comments on the support required by the underlying system for this architecture.

2. Architectures

An architecture provides a structure through which a large or complex system can be understood and reasoned about. Weaknesses can be identified before the system is built. In creating this structure, the system architect chooses to represent a set of components and various connections between the components while abstracting away other details of the system, thus forming a particular perspective. Different choices of components or connections will provide different architectures, and therefore different perspectives, of the system.

For any system type (e.g., office buildings, highways, information systems) there are usually a small number of architectural styles, each with its own strengths and weaknesses, from which to choose. A style determines a vocabulary of components and connectors that the architect uses to define an instance of that style, along with constraints on their combinations [4]. The designer of a new system usually starts with a style for which the strengths best meet the needs of the system, perhaps selecting one that was successful for a similar system.

An information system architect can specify several complementary architectures, each providing a perspective on a different concern. A *system architecture* describes the hardware components and the communication channels that connect them to form a distributed or networked system. A *software architecture* [4] describes the computational components and the information and/or control flows between them; styles include pipes and filters, object based, and layered. These two architectures are somewhat independent in that many software components may reside on a hardware component, or a single software component may span many hardware

components (although performance and fault-tolerance considerations normally strongly influence the mapping between the architectures). A third architecture, the *information architecture*, gives an information-centric view of the system: its components are collections of information, and its connections define the ways in which these collections relate to each other.

The DGSA defines an information architectural style (or generic architecture) in which the need to protect and to share information determines conceptual components, called *information domains*, and constraining connections between them. The DGSA components are independent of components of the system and the software architectures. For example, a server in a client-server software architecture might manage objects that require different protections and must therefore be in different information domains. Alternatively, several servers might manage objects of different types that are used together under a single protection policy in a single information domain. Likewise, information domain boundaries do not necessarily correspond to hardware platform boundaries.

3. Security Architectures

The desired protection of information and the need to share that information determine a DGSA-style architecture. We first define information domains and their connections. We then demonstrate their use through an example. Finally, we discuss the control of management information and other implementation concerns.

3.1. Information Domains

Information domains partition the information in the system such that all information in a domain has the same protection requirements—there are no security-relevant distinctions among information in a domain. The properties of the information that the system security policy uses to define the protection requirements, referred to as *security attributes*, help define the domains; each domain has a specific value for these attributes.

The need to share information further refines the domains by requiring that a set of *principals* (referred to as *users* in [2]) share all of the information in a domain. A principal may correspond to some individual, a cryptographic key, a principal acting in a role, a conjunction of principals, or a principal acting on behalf of another principal [1]. As with information, the characteristics of principals that the system security policy uses to define requirements are referred to as security attributes; the value for a principal determines if it belongs to a particular domain and how it may access data in that domain.

The transfer of information from one domain to another forms a connection between components. A principal can make a transfer if the system security policy allows it to share information in both the source and the destination domain and to make the transfer. Thus, the security policy defines the constraints on the connections. If the security policy forbids an information transfer from one principal to another, there can be no transfers from a domain in which the first principal modifies information to one in which the second can observe information.

The security requirements that pertain to the information in a domain, along with the need to share that information, are used to derive a *domain security policy*. This policy specifies which principals may operate in the domain and in what ways each of these member principals may access the information. The accesses allowed may include ways to modify or observe the information, or the transfer of information to or from the domain. The domain security policy may also specify identification and authentication requirements for the principals and auditing requirements for accesses. Finally, the policy may specify Quality of Service requirements.

Information domains as described above are similar to Distributed Compartments [5], and much of the process that we describe in subsequent sections would apply to a system with this security architecture. One difference is that DGSA information domains have no a priori structured relationships. The relationships created by the ability to transfer information might form a partial order, but they also might have cycles or be nontransitive. Also, while the ability to transfer information between two domains implies that the domains have at least one principal in common, the principals in the destination domain need not be a subset of those in the source domain.

3.2. Example System

Consider an information system that a university uses to maintain grades for various courses. The security policy for this system is that professors may enter or modify scores for assignments and exams, record attendance, and calculate course grades for those courses that they teach. Individual students may observe information about themselves, but not about other students. A recorder (which is internal to the system and does not correspond to any external individual) maintains the distribution of grades for each course, which anybody may observe. Thus, the security attributes for assignments, attendance, and course grades are the professor that teaches the course and the student to which it pertains. Because the security policy does not distinguish among the grade distributions, they share a single attribute value. The principals in the system are the professors, the students, and the recorder, each with an identity attribute. There must be a scores information domain for each professor-student pair containing the scores, attendance, and course grades assigned by the professor to that student. There also would be a distribution domain for the grade distributions and a separate personal domain for each professor and each student. The security policy for a scores domain specifies that the professor may observe and modify information in the domain, the student and the recorder may observe information, and no other principals have any access. The policy might also specify that additions and changes by a professor and observations by the recorder must be audited.

Information sometimes must be available in multiple domains, in accordance with the system security policy. In the grading system, test grades that a professor calculates in his personal domain must be available in the appropriate scores domains after grading is complete, and grades from the scores domains must be available in the distribution domain. Information in one domain can be transferred to another domain either by copying it (information added to the distribution domain remains in the scores domain also) or by

moving it (after a grade has been placed in a scores domain, it is no longer needed in the professor's personal domain). In either case, the transferred information assumes the security attributes of the domain to which it is transferred.

A transfer is accomplished by a principal acting in both the sending and the receiving domains. Thus, moving a test score from a professor's personal domain to a scores domain requires that the professor act in both domains. As with any other operation on the information of a domain, the security policy for the sending and the receiving domains must permit the transfer by the principal making the transfer. The security policy for a scores domain must allow the professor associated with that domain to receive information from the professor's personal domain, and it must allow the recorder to copy information to the distribution domain. Figure 1 shows the domains for the grading system and the routes by which information may be transferred between them.

3.3. Management Information

A Management Information Base is the information used to manage an information domain; the security-related portion is referred to as a Security Management Information Base (SMIB). A SMIB may include the security attribute values for the information in the domain, the principals that are members of the domain (possibly using the values of the security attributes of those principals), authentication and auditing requirements, and access control restrictions for each of the various principals. Principals are added to or removed from a domain by modifying the SMIB for that domain.

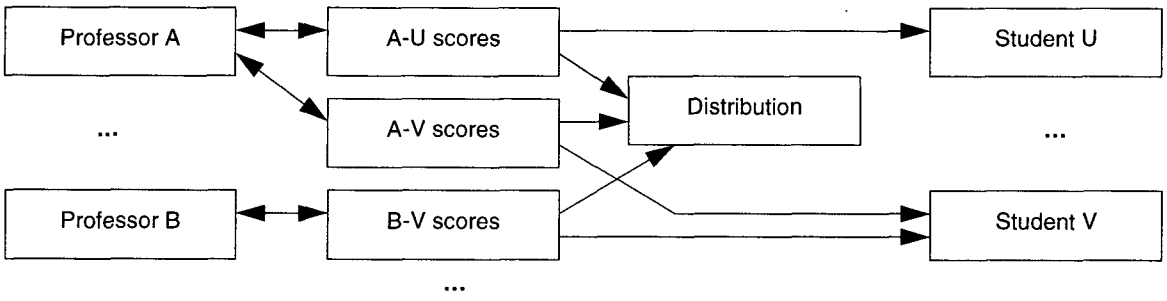
Because a SMIB is information, it is contained in one or more information domains that control access to it. There are two options for how the domains containing a SMIB relate to the domains that the SMIB describes:

- The SMIB is part of the domain that it describes. Thus, the security policy for the management information is the same as for the rest of the information in the domain.
- The SMIB is not part of the domain that it describes, but is instead in separate domains with security policies that differ from those of the domains to which it refers. (This leads to a recursive situation because there must now be a SMIB for the separate domains: eventually for some domain the first option will be used.)

For the university grading system, the SMIB contains the security attributes for each of the information domains and the set of principals permitted to access the domains. The security policy might be that only a registrar principal may modify the SMIB. Because this policy differs from those of the domains already described, it must be in a domain different from those shown in Figure 1 (the second option). The SMIB for this new domain is also only accessible by the registrar and therefore is contained in itself (the first option).

The collection of information domains in a system is frequently not static but changes over time. For example, students may add and drop classes. To create or destroy a domain, SMIB entries for that domain are created or destroyed. Thus, the security policies for the domains containing the SMIB control what principals may create and destroy domains. The security policy for the university SMIB domain permits the registrar to create and destroy SMIBs.

The security policy related to some information may change over time. A coalition or business alliance might be forged that requires wider dissemination of plans, or a document might be finalized so that further modification must be forbidden. These changes are



Professor P	P-S scores	Distribution	Student S
principal: P policy— P may: observe, modify, transfer to/from P-S scores	principals: P, S, Recorder policy— P may: transfer to/from Professor P; S may: copy to Student S; Recorder may: copy to Distribution	principals: Recorder, A, B, U, V, ... policy— Recorder may: observe, modify, copy from P-S scores; P,S may: observe-distribution	principal: S policy— S may: observe, modify, copy from P-S scores

Figure 1. Domains for the Grades Example

made to the policy representation in the SMIB and are controlled by the policy pertaining to the SMIB.

3.4. Implementation Concerns

The DGSA requires that information systems support a protection strategy of *strict isolation.* Except when the security policy allows information transfers, information domains must be isolated from each other. Because different domains have separate security policies, information must not be inferable in domains other than the one to which it belongs. In the university grading system, a professor's activities in the scores domains for different students in his classes must be isolated from each other.

A set of security services is used to isolate information domains and to enforce their security policies. The DGSA defines these services to be authentication, access control, data integrity, data confidentiality, non-repudiation, and availability. Security management functions, including audit and key management, support these services. These services come in a range of strengths, with stronger versions usually incurring greater overhead or difficulty in use. For example, authentication ranges from short user-generated passwords to cryptographic cards to biometrics. The strength of service for each of the domains depends on the domain security policy, the sensitivity of the information, and the environment in which the system resides. In the grades example, the integrity of the data in a scores domain is very important (future employment for the student may depend on the values), while availability is not a major concern. Alternatively, availability to a student's personal domain may be very important if it contains an assignment that the student must complete by that afternoon.

4. Information Domains and System Architectures

The previous description of information domains is independent of any hardware configuration. Information domains are defined by the needs to share and to protect information without concern for the location in the system of either the principals or the information. Principals and the information that they are using might be on the same hardware component, or they might be located on different continents and connected by a public network. The grades system could be implemented entirely on a main frame, or the professor and student personal domains could be on separate personal computers with the scores and distribution domains on a central server. The security architecture is the same in both cases. We will refer to the components of a system architecture as *end systems* and the connections as a *communication network.*

The combination of a security architecture and a system architecture forms a new *sited-domain* architectural perspective that is a refinement of both the security and the system architectures. A component of this architecture is a sited domain (Figure 2), consisting of the information from an information domain of the security architecture that is present on a particular end system of the system architecture (information in transit is assumed to remain on the source end system until it is received by the destination). The principals of a sited domain are those that are members of the information domain and are operating on the end system. Note that an association is required between the representation of principals in the domain security policy and the principals operating on the end system, possibly using handles [5]. The security policy for a sited domain is the same as that of the information domain from which it is derived. Thus, the information in an information domain is the union of the information in the sited domains derived from it, and the members of an information domain are the union of those in the sited domains. Note that a piece of information from a domain might be replicated or otherwise represented on multiple end systems, so that the pieces of information in a domain do not necessarily exactly match the pieces of information in the sited domains derived from it.

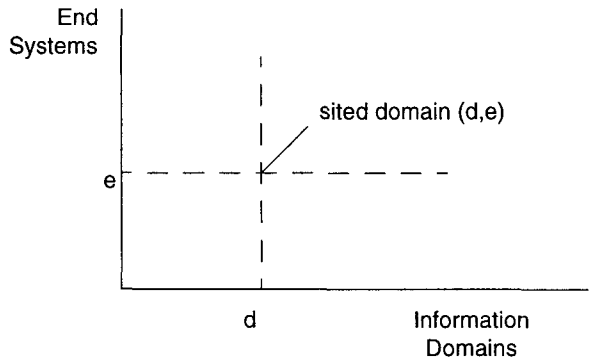


Figure 2. Security-Hardware Architecture

The DGSA makes two important restrictions on sited-domain architectures. The first is that connections occur only along the Information Domain or the End System axis. Thus, information may be transferred between end systems within a domain, or between domains on a single end system. This restriction is due to a general lack of security services provided by the communication networks that connect the end systems. In the grading system, assume that the student personal domains are on personal computers and the scores domains are on a common server. The restriction implies that copying scores to a student domain must be done either on the server, requiring that the student domain also contain a sited domain on the server, or on the student's personal computer, requiring that the scores domain have a sited domain on the personal computer.

The second restriction is that each end system supporting a domain must adequately provide the required security services. Also an end system supporting more than one information domain must be able to provide strict isolation for those domains. For example, as argued above, strong data integrity must be provided for the scores domain. Because a student's personal computer cannot be trusted to prevent the student from altering scores, the sited domain (scores, personal computer) must be empty. Further, the personal computer does not have the services needed to provide strict isolation between domains. Thus, the scores domain cannot be represented on a student's personal computer and copying from it to a student personal domain is represented in the sited domain architecture as

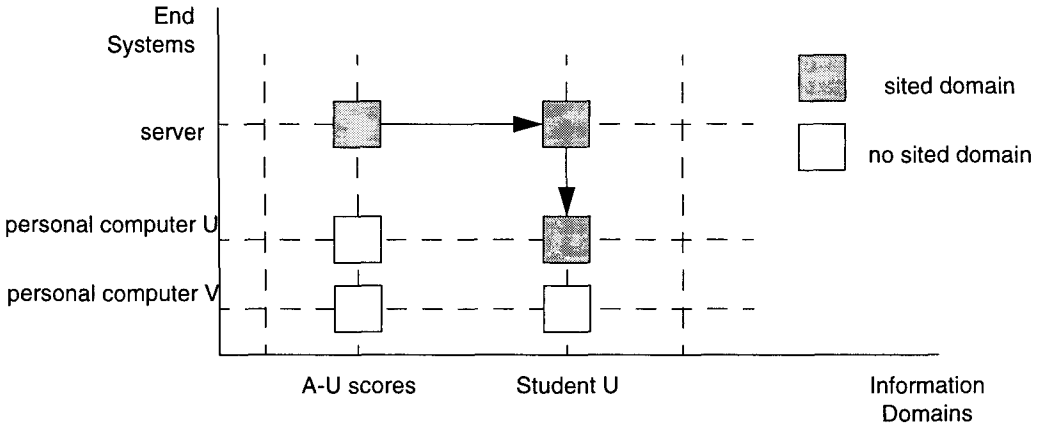


Figure 3. Refinement of copy from A-U scores to Student U

a copy from (scores, server) to (personal, server), possibly followed by a move to (personal, personal computer), as shown in Figure 3. Security services in a strength sufficient for each student's personal domain must be provided by the common server. In particular, while authentication on the student's personal computer can be provided by physical means (a locked dorm room), a password or other method must be used to gain entry to sited domain (personal, server).

The flow of information between domains internal to an organization and those external to that organization is frequently controlled by a firewall. This is an end system containing both internal and external domains, along with a security policy regulating the flow between them. The firewall must provide the security services required by any of the domains, but the internal end systems do not need to provide support for the external domains (external domains do not extend inside the firewall), and the external systems are not relied on to provide services in support of internal domains (internal domains do not extend outside the firewall).

The sited domains supporting a common information domain on different end systems must maintain the protections required by the information domain security policy while communicating with each other. A *security association* is the totality of communications and security mechanisms and functions that securely binds together these sited domains [2]. A security association must ensure that:

- the destination end system is accredited to handle information of the sensitivity and category that the source may provide,
- the identical security policy is enforced by the sited domains,
- data confidentiality and data integrity are maintained during transit,
- the policy availability constraints are satisfied, and
- accountability for the transit is maintained, if required by the security policy.

Frequently, the communication network cannot ensure confidentiality, integrity, and accountability during transit. The end systems must cooperate to provide these services.

5. Information Domains and Software Architectures

Just as the description of information domains is independent of the system architecture, it is also independent of the software architecture. A principal may need to simultaneously access information of different types, each managed by a different object manager (also called a server in a client-server architecture). Thus, different components of the software architecture, such as test scores and attendance records in the grading example, may coexist in an information domain. Conversely, software objects such as files or test scores, implemented by a single manager, will be in different information domains if the security policy requires that they be protected differently. Note that a manager may exist on several end systems, and an object that it manages may be replicated on several of those end systems.

The combination of a security architecture and a software architecture forms a new *managed-domain* architectural perspective that is a refinement of both the security and the software architectures. A component of this architecture is a managed domain (Figure 4), consisting of the information from an information domain of the security architecture that is represented as objects by a particular manager or server of the software architecture. The principals of a managed domain are those that are members of the information domain and are permitted accesses supported by the manager. The security policy for a managed domain is the same as that of the information domain from which it is derived. Thus, the information in an information domain is the union of the information represented by objects in the managed domains derived from it, and the members of an information domain are the union of those in the managed domains.

Intradomain security policies are enforced by the managers (or by *wrappers* applied to them) that define the effects of various opera-

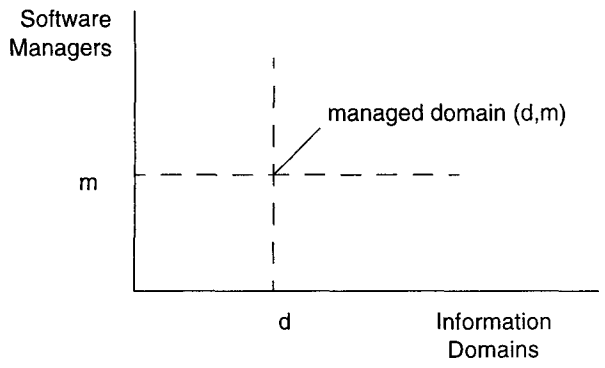


Figure 4. Security-Software Architecture

tions. Enforcement of an integrity policy, permitting a particular principal to observe but not modify information in an information domain, depends on knowing that the implementation of an update operation modifies information and therefore that execution of update by that individual is a security violation. Note that the information in a managed domain includes the manager's private state, in that this state can affect future actions performed by the manager; strict isolation requires that state changes that result from an operation in one information domain must not affect other information domains.

While all objects representing information in a domain must have the same security attributes, the different managers may each implement different sets of operations and therefore require different permissions for the principal that attempts to invoke those operations. Thus, in the distribution domain of the grading example, there will be scores objects copied from a scores domain and managed by a scores manager, in addition to the distribution objects. The observe-distribution operation is not implemented by the scores manager and therefore the professors and the students are unable to access scores objects in the distribution domain. Similarly, the modify operation might have different semantic meaning for scores or distribution objects.

The ability of a manager to cross information domain boundaries means that moving information between domains need not be computationally expensive. The information does not need to be physically moved or reformatted. All that needs to be done is to change the security attributes associated with it to reflect the protection provided to the data in the new domain. The file manager is trusted on most systems to enforce different sets of protections on different files, and the protection given to a file can be changed (the file moved to a different domain) by changing the owner or the protection bit fields. However, a manager that crosses domain boundaries must be trusted to maintain strict isolation between those domains, as described in Section 3.

Frequently, code that is not trusted either to enforce the security policy or to maintain strict isolation must be used. Such code must be completely contained within a domain such that any principals that are permitted to execute it have all permissions within the domain, except perhaps to transfer information into and out of the domain. The only interaction allowed between this code and code in other domains is through controlled domain transfers. The damage that can be done by such code is thus isolated to the domain, and transfers between this and other domains are tightly controlled. In the student grading system example, all code that modifies scores or calculates grades is run in a professor's personal domain on behalf of a pseudo-principal that is not allowed to transfer information into or out of the domain. Before the code is run, the professor transfers the minimal collection of information into the domain. Upon completion, the professor checks the results before transferring them to a scores domain.

6. Using the Architecture

A three-dimensional information system architecture has been presented, of which security is one independent dimension (Figure 5).

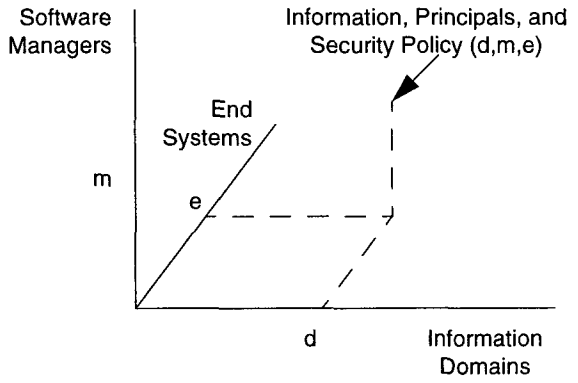


Figure 5. 3-Dimensional Information System Architecture

Each component of the architecture is the information from one domain (**d**) that is represented on an end system (**e**) and managed by a particular software manager (**m**), together with the principals that are members of **d** and active on **e**, subject to the security policy of **d** as interpreted by **m**. This architecture must next be translated into an implementation.

A computation on a system constructed using the concepts presented here occurs in a domain containing the minimal amount of information necessary. Confidentiality is maintained by limiting the set of individuals that belong to a domain and by controlling to which other domains results can be transferred. Integrity is maintained by limiting the sources of information that may be transferred into a domain. Nontransitive security policies, in which allowing information flows from *a* to *b* and from *b* to *c* does not necessarily allow information to flow directly from *a* to *c*, can easily be established; this is currently a difficult problem for systems based on a hierarchical security structure. Untrusted software can be safely executed within a domain to which required information is copied and from which the results are tightly controlled.

The underlying system support for information domains must ensure strict isolation. All interaction between principals must occur as the result of changes to information in a domain to which they all belong. What is needed are separate address spaces for each information domain, connected by channels that are subject to a transfer policy enforcement mechanism. To the extent possible, system resources such as files, printers, etc. should be permanently assigned to a domain. Within each domain address space, principals can be represented as threads of control. Entry by a principal into the domain is represented by thread creation and exit by thread destruction. However, unlike most operating systems with light-weight threads, these threads must be separately identified by a set of attributes. These attributes are checked against the security policy whenever the thread issues a command for service, and are sent with any transfer request. The underlying system must also guarantee any availability requirements of the security policy in its scheduling of the threads.

Enforcement of security occurs in the implementation of access to the information. When information is transferred to a new domain, a different security policy is applied to that information. This policy change can be managed by separating policy definition from the enforcement, defining a Security Policy Decision Function for each information domain. A large class of security policies can be represented in this manner [7].

Security Associations are required between end systems that mutually support a domain. These associations must establish an agreement on the protocols and protections to use during communications, the policy to be enforced by each end system, and the strength of services required for that enforcement. In order for an association like this to work, there must be a trust relationship between the end systems that each will honor the agreement.

Current commercial systems do not provide the needed support for the architectures developed using the methods described in this paper. The required isolation has not been a design goal for current systems, and consequently most are inadequate. Also, either their security mechanisms are transitive, or they are discretionary and therefore overall properties of information flow are hard to guarantee. As the use of mobile code increases, we hope to see better isolation mechanisms that supply the needed support.

Acknowledgements

This work was supported by the National Security Agency under contract DASW01-98-C-0067; the views expressed are those of the author and do not reflect the official policy or position of the U.S. Government or the Department of Defense. Many of the ideas presented are the result of collaboration with Ed Feustel on designing information systems for a complex and dynamic policy environment, and to discussions with Terry Mayfield. Finally, a special thanks is due to the attendees at the NSPW for the lively discussion.

References

[1] Martín Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. *ACM Trans. on Programming Languages and Systems 15*(4): 706–734, September 1993.

[2] Defense Information Systems Agency, Center for Standards. *Department of Defense (DoD) Goal Security Architecture (DGSA),* Version 3.0, April 1996. Volume 6 of *Department of Defense Technical Architecture Framework for Information Management (TAFIM).*

[3] Edward A. Feustel and Terry Mayfield. The DGSA: Unmet security challenges for operating system designers. *Operating Systems Review 32*(1): 3–22, January 1998.

[4] David Garlan and Mary Shaw. An introduction to software architecture. In *Advances in Software Engineering and Knowledge Engineering*, edited by V. Ambriola and G. Tortora, World Scientific Publishing Company, 1993.

[5] Steven J. Greenwald. A new security policy for distributed resource management and access control. In *Proceedings of the New Security Paradigms Workshop*, pages 74-86, Lake Arrowhead, CA, September 1996.

[6] Tom Lowman and Douglas Mosier. Applying the DoD Goal Security Architecture as a methodology for the development of system and enterprise security architectures. In *Proceedings of the Thirteenth Annual Computer Security Applications Conference*, pages 183-193, San Diego, CA, IEEE Computer Society, December 1997.

[7] Duane Olawsky, Todd Fine, Edward Schneider, and Ray Spencer. Developing and using a "policy neutral" access control policy. In *Proceedings of the New Security Paradigms Workshop*, pages 60-67, Lake Arrowhead, CA, September 1996.

[8] Edward A. Schneider, Edward A. Feustel, and Ronald S. Ross. *Assessing DoD Goal Security Architecture (DGSA) Support in Commercially Available Operating Systems and Hardware Platforms.* IDA Paper P-3375, November 1997.