Confidentiality, Integrity, Assured Service: Tying Security All Together

Grace L. Hammonds
AGCS, Incorporated
91 Montvale Avenue
Stoneham, Massachusetts 02180

Abstract

The purpose of this paper is to explore the commonalities in the three security areas, and to offer a framework that not only relates all three, but also can be used to distinguish security problems that are easily addressed from those that cannot. The goal is to have a basis for a consistent level of analysis and, eventually, consensus on what constitutes viable solutions for enhancing the overall security in our information system.

1 Introduction

Over the years, studies in the foundations of information security have tended to focus on one or more of the following policy areas:

- *Confidentiality*, the protection of data from compromise, of special interest to the U.S. Department of Defense (DoD);
- *Integrity*, the prevention of improper modification, a major concern of the private sector;
- *Assured Service* (also referred to as *availability*), the prevention of denial of service.

There are a number of reasons for this [?]: different communities of interest have different priorities; there, is a lack of consensus on the meaning of basic terms; there is a little agreement on appropriate solutions in hardware or software or both. *Assured service* is viewed, in the extreme, as impossible to obtain since it implies absolute reliability and fault tolerance. However, it is difficult, if not impossible, to address any one area without acknowledging the nececessity of addressing the others to some degree.

Today, we have accepted notions for confidentiality, now codified in various evaluation criteria and government regulations. Indeed, commercial trusted computer systems have begun to treat confidentiality based on these notions. One can also find numerous studies on the various aspects of *integrity*, but there are as yet few conclusions on how to address it [?]. The *assured service* area has seen far less review (but see [?]), however, there is general agreement that it is an important consideration that should be addressed.

A number of studies have begun to address how to tie the three security policies together. McCumber [?] shows high level relationships among the three, by mapping the transmission, storage, and processing of information to general categories of safeguards-technology, policy and practices, and education, training, and awareness. An NCSC technical report [?] examines in depth a number of shades of integrity and relates it to the availability of system services (i.e., system integrity). Hosmer [?] shows how different or incompatible policies might be supported within the same information system.

The next three sections identify general models for each of the three security areas: confidentiality, integrity, and assured service. Although the confidentiality model is not new, the integrity and denial of service models are original. The paper concludes with the "tying" together of the three models.

2 Confidentiality Model

An often used view of confidentiality is the "Reference Monitor" (RM) model, shown below in Figure 1 [?]. This model has provided the basis for many trusted computer system developments for protecting classified data from compromise. In Step 1, a subject (e.g., task or process) attempts to access an object (e.g., file, memory), causing the RM to be invoked. In Step 2, the RM attempts to validate the access using a set of rules (generally embodied in its instructions and algorithms) and data containing security attributes of the subject and object. If the check is successful, then in Step 3, the access is permitted. Otherwise it is disallowed.

Mechanisms that implement this model of security protection can generally be localized in the RM component, that has three basic attributes.

1. Complete—always invoked to perform the checking;

2. Verifiable—small and simple to permit analysis and testing;

3. Tamperproof—able to protect itself from modification.

Figure 1 is significant not only because of the components and relationships it illustrates, but also because of the "flow" of information and control implied by the *arrows*. In other words, the arrows can be thought of as subjects and objects. This idea will be pursued below.

The RM model has been extended to integrity and assured service in limited ways. Integrity protection falls under this model to the extent that the RM can validate the right of a subject to modify or to append to an object. Because the security data can consist of one or more objects, its validity and correctness must be protected, and only authorized users should have the right to change the data. Assured service is an issue in the self-defense aspects of the RM: the Reference Monitor should be designed to ward off active attacks through software that would hinder its normal operation.

3 Integrity Model

In recent years, a number of papers have been written that analyze different aspects of integrity, one of the more comprehensive being the NCSC technical report "Integrity in Automated Information Systems." That report identifies a number of aspects of integrity and presents a framework for their discussion and analysis.

In this paper, we offer an alternative framework in Figure 2. Figure 2 shows the components of the integrity model and their relationships in an information system. *Information system* is used very broadly and can include users, data, system software, applications (e.g., DBMS), computer hardware, and even a network with distributed applications. The components are of two types, depending on their respective roles. An *initiator* component performs an action on a target component. In Step 1, the Change Agent (e.g., user, application, operating system, hardware) affects a system Resource, such as a append to a file, or update a database record. In Step 2, the modification has the result Resource*t*.

For example, if the Change Agent is appending to an object, Resource*t* is a superset of Resource. If the Change Agent performs a duplication, Resource*t* is a new object with the same *value* as Resource.

The solid arrow in the diagram indicates cause and effect: an action taken on a component that causes a change to system resources. The dashed arrow represents the change, giving a new value. In this discussion, the notion of *change* includes the normal idea of "modification of data" as well as the concept of "command"—when a user enters commands, it causes the system to perform an action, and thereby *changing*, or adjusting, its behavior to perform what is requested. These actions could result in integrity problems, that is, negative effects. The effect of the action is manifest in the target in ways that are dependent on the Resource type.

As in confidentiality, the solid arrow can also be viewed as an information flow—transition of commands and data to resources.

The
Change Agent may have multiple instantiations—for instance a *user* creates a *SQL program* that invokes a *database management system* which in turn relies on the underlying *operating system* to change data in a disk partition.

Given this model, integrity policy, integrity mechanisms, and assurance of integrity can be defined as follows. An *integrity policy* addresses the valid conditions under which an initiator may legitimately affect a target. More precisely, an *integrity policy* identifies

- Constraints on an initiator, the "cause," to prevent an action that could result in an *integrity problem*, that is, a loss of integrity. An example of such constraints could be requiring a user to be *authorized* to perform the action. Another example would be requiring users to have a controlled command set, say, through a database management system or menu-based interface.

- Constraints on a target to prevent a deleterious effect from taking place. For instance, in a database, a requirement that the database not be updated unless and until a transaction be complete.

Integrity mechanisms are, protective measures applied to prevent or minimize an integrity problem.

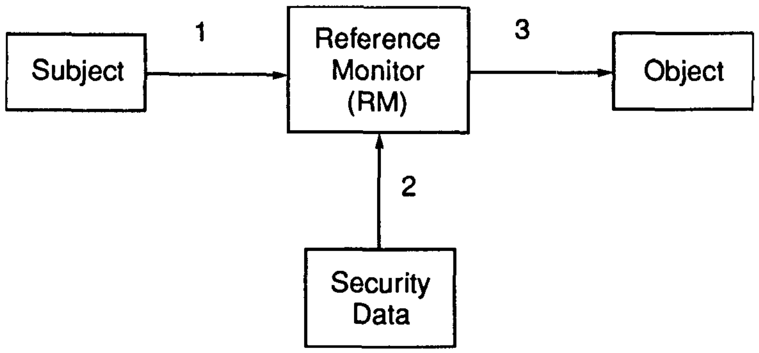

Figure 1: A reference monitor

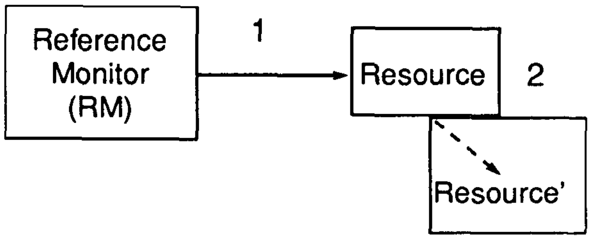

Figure 2: Components of an integrity model

Such measures can provide

- Detection of attacks, attempts to circumvent controls;
- Controls and restrictions to avoid or reduce the risk of a problem;
- Detections of problems after they have occurred;
- Recovery measures that attempt to undo the problem, addressing the loss of integrity when problems cannot be prevented (i.e., to correct or mitigate any problems).

Unlike in the Confidentiality Model, implementations of mechanisms for integrity cannot in every case be localized in a certain component or components. Integrity problems may arise through faulty initiator requests or through inappropriate or incorrect processing.

Assurance of integrity is the evidence that within an information system.

- An initiator is appropriately authorized for any effects it causes;
- The change of state from Resource to Resource/ is correct or has sufficient quality. "Quality" is context-specific.

4 Assured Service Model

Assured service issues are often discussed in the context of integrity because loss of integrity (bad data) can result in a subsequent loss of service (misdirected code). Assured service also includes *completion* and *continuity* of processing, in a correct as well as a timely way. In other words, assured service means that an information system does *what* you want, *when* you want it.

In the assured service model, shown in Figure 3, there are two components, an Initiator and a Processing Component. The Initiator actively seeks die services of the processing component If successful, the service is performed expected. If the action is inhibited, the service is not performed as expected. For example, a user can interact with an application, or system software may invoke one of the system utilities.

As in the Integrity Model, the solid arrow represents cause and effect, as well as the flow of information and control. The difference is the notion of *time* as an attribute of an effect. For instance, the Processing Agent is delayed from completing an action for an unacceptable amount of time. If certain processing is delayed a few hours, that might be unacceptable, a delay of a few seconds could fall within tolerance.

Using this model, we have a framework for identifying assured service policy as the provision of a *suffi-*

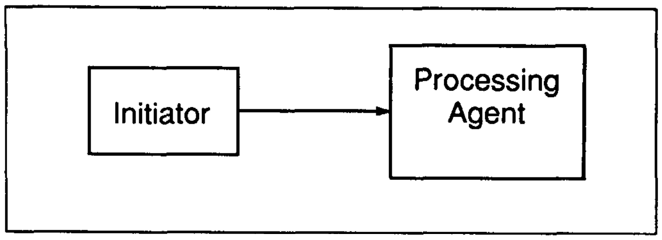

Figure 3: An assured service model

cient, or acceptable, level of service. Sufficient service is assured if the

1. Initiator is authorized to invoke a Processing Agent in the desired manner (an unauthorized access is a *theft of service*);
2. Initiator invokes the Processing Agent requested, and not a bogus one;
3. Initiator invokes the minimum necessary Processing Agents to achieve the desired effect.

Mechanisms to support assured service include

1. Detection of attacks, attempts to affect processing;
2. Controls and restrictions to avoid or reduce the risk of a problem (e.g., backup and redundant processing agents);
3. Detection of problems that are unavoidable: breakdowns and slowdowns;
4. Recovery measures in the event of a problem.

Again, unlike in the Confidentiality Model, implementations of mechanisms to support assured service cannot in every case be localized in a certain component or components. Such problems may arise through faulty initiator requests, or through inappropriate or incorrect processing.

5 Security Framework

The proposed combined Information System security framework for confidentiality, integrity, and assured service is shown in Figure 4.

The components in the figure are the

- **Initiator**, possibly a user or process (Subject in the Confidentiality Model, Change Agent in the Integrity Model, Initiator in the Assured Service Model). In Step 1, the Initiator begins an action.
- **Processing Agent**, that performs an operation (Step 3) on behalf of the initiator. Examples include an application and the operating system. (The Processing Agent is the RM in Confidentiality Model; it is implied in Integrity and Assured Service models.) This component may be decomposed, recursively, into a series of Processing Agents. The Processing Agent may have a component that validates access rights, the Reference Monitor, as in the Confidentiality Model.
- **Reference Data**, may be used by the Processing Agent to perform its actions (Step 2). The Reference Monitor may have its own private set of data for validating the request.
- **Resource**, target of the operation (Object in Confidentiality Model, Resource in Integrity Model, Processing Agent in Assured Service Model). The Resource is accessed in a manner dependent on its type and on the requested operation—read, or modify, or execute.
- **The Resource′**, the result of the operation in the event a change takes place.

In step 1, an Initiator invokes a service through a Processing Agent that, depending on the service, may include a Reference Monitor check of the rights of the Initiator. In any event, data is available to the Processing Agent (step 2) to support its processing. If the processing is authorized, the service takes place (step 3), which may include the invocation of other processing agents in sequence. Step 4 is the transition to the new state of the Resource, Resource′.

Given this framework, aside from the obvious mappings from the three separate security models, this framework allows us now to view the three types of security with a common set of assumptions and policy considerations. For example,

- The solid arrows, representing requests (or commands) and perhaps data, must also be protected

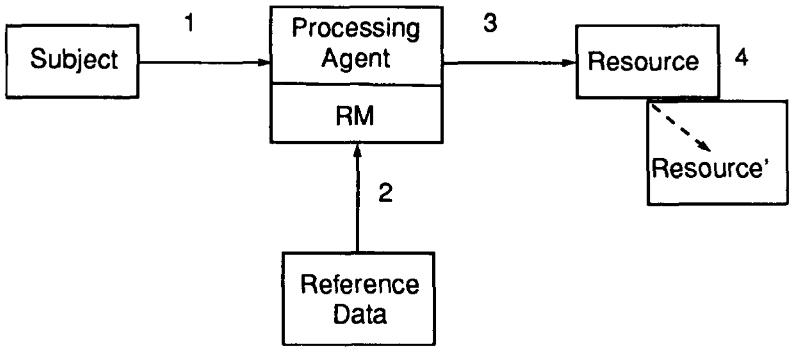

Figure 4: Security framework

as objects. It could embody the notion of the subjects identification and forms of "trusted path."

- Threats to an information system may be targeted at any of the components (and arrows) of the model, including at (or after) the creation of Resource'.

- A read operation (as occurs under the Confidentiality Model) carries the risk of the return of an incorrect view (Resource' could be the image returned).

- A Processing Agent may take the role of an Initiator, Resource or Change Agent at different points in time.

- Integrity and Assured Service are often confused because the Resource may be either a data object or Processing Agent for a later set of operations.

- Security problems of all types may be caused by not simply a single event, but by a series of events. For instance an attack of data used later by a Processing Agent component, can cause and integrity problem in Resource' at another point in time.

- One aspect of Assured Service can be addressed by considering the attribute *time* as a Resource (e.g., the clock, or the time slice), to be protected to the same extent any other resource can be protected.

- The Reference Monitor concept is only useful insofar as one can identify attributes of initiators and targets that can be validated under some predefined set of rules, and within a predefined series of actions.

6 Next Steps

The combined security framework described here can be mapped to the considerable body of existing work on Integrity and Assured Service. This mapping will be the subject of future work in this area. The more important aspect of this model is its use in understanding what has heretofore been considered intractable problems. These extensions are under study as well.

References

[1] Gasser, M., *Building a Secure Computer System*, Van Nostrand Reinhold, 1988.

[2] "Integrity in Automated Information Systems," National Computer Security Center, C Technical Report 79-91, September 1991.

[3] "Assured Service Concepts and Models," Volumes 1-3, Secure Computing Technology Corporation, Technical Report, Contract Number F30602-90-C-0025, October 1991.

[4] McCumber, J.R., "Information Systems Security: A Comprehensive Model," *Proceedings of the 14th NCSC Conference*, October 1991.

[5] Hosmer, H.H., "The Multipolicy Paradigm for Trusted Systems."

[6] Anderson, J. P., Computer Security Technology Planning Study, ESD-TR-73-51, Volume 1, ESD/AFSC, October 1972.