A Comparison of Commercial and Military Computer Security Policies

David D. Clark* - David R. Wilson**

* Senior Research Scientist, MIT Laboratory for Computer Science
545 Technology Square, Cambridge, MA 02139
** Director, Information Security Services, Ernst & Whinney
2000 National City Center, Cleveland, OH 44114

ABSTRACT

Most discussions of computer security focus on control of disclosure. In particular, the U.S. Department of Defense has developed a set of criteria for computer mechanisms to provide control of classified information. However, for that core of data processing concerned with business operation and control of assets, the primary security concern is data integrity. This paper presents a policy for data integrity based on commercial data processing practices, and compares the mechanisms needed for this policy with the mechanisms needed to enforce the lattice model for information security. We argue that a lattice model is not sufficient to characterize integrity policies, and that distinct mechanisms are needed to control disclosure and to provide integrity.

INTRODUCTION

Any discussion of mechanisms to enforce computer security must involve a particular security policy that specifies the security goals the system must meet and the threats it must resist. For example, the high-level security goals most often specified are that the system should prevent unauthorized disclosure or theft of information, should prevent unauthorized modification of information, and should prevent denial of service. Traditional threats that must be countered are system penetration by unauthorized persons, unauthorized actions by authorized persons, and abuse of special privileges by systems programmers and facility operators. These threats may be intentional or accidental.

Imprecise or conflicting assumptions about desired policies often confuse discussions of computer security mechanisms. In particular, in comparing commercial and military systems, a misunderstanding about the underlying policies the two are trying to enforce often leads to difficulty in understanding the motivation for certain mechanisms that have been developed and espoused by one group or the other. This paper discusses the military security policy, presents a security policy valid in many commercial situations, and then compares the two policies to reveal important differences between them.

The military security policy we are referring to is a set of policies that regulate the control of classified information within the government. This well-understood, high-level information security policy is that all classified information shall be protected from unauthorized disclosure or declassification. Mechanisms used to enforce this policy include the mandatory labeling of all documents with their classification level, and the assigning of user access categories based on the investigation (or "clearing") of all persons permitted to use this information. During the last 15 to 20 years, considerable effort has gone into determining which mechanisms should be used to enforce this policy within a computer. Mechanisms such as identification and authorization of users, generation of audit information, and association of access control labels with all information objects are well understood. This policy is defined in the Department of Defense Trusted Computer System Evaluation Criteria [DOD], often called the "Orange Book" from the color of its cover. It articulates a standard for maintaining confidentiality of information and is, for the purposes of our paper, the "military" information security policy. The term "military" is perhaps not the most descriptive characterization of this policy; it is relevant to any situation in which access rules for sensitive material must be enforced. We use the term "military" as a concise tag which at least captures the origin of the policy.

In the commercial environment, preventing disclosure is often important, but preventing unauthorized data modification is usually paramount. In particular, for that core of commercial data processing that relates to management and accounting for assets, preventing fraud and error is the primary goal. This goal is addressed by enforcing the integrity rather than the privacy of the information. For this reason, the policy we will concern ourselves with is one that addresses integrity rather than disclosure. We will call this a commercial policy, in contrast to the military information security policy. We are not suggesting that integrity plays no role in military concerns. However, to the extent that the Orange Book is the articulation of the military information security policy, there is a clear difference of emphasis in the military and commercial worlds.

While the accounting principles that are the basis of fraud and error control are well known, there is yet no Orange Book for the commercial sector that articulates how these policies are to be implemented in the context of a computer system. This makes it difficult to answer the question of whether the mechanisms designed to enforce military information security policies also apply to enforcing commercial integrity policies. It would be very nice if the same mechanisms could meet both goals, thus enabling the commercial and military worlds to share the development costs of the necessary mechanisms. However, we will argue that two distinct classes of mechanism will be required, because some of the mechanisms needed to enforce disclosure controls and integrity controls are very different.

Therefore, the goal of this paper is to defend two conclusions. First, there is a distinct set of security policies, related to integrity rather than disclosure, which are often of highest priority in the commercial data processing environment. Second, some separate mechanisms are required for enforcement of these policies, disjoint from those of the Orange Book.

MILITARY SECURITY POLICY

The policies associated with the management of classified information, and the mechanisms used to enforce these policies, are carefully defined and well understood within the military. However, these mechanisms are not necessarily well understood in the commercial world, which normally does not have such a complex requirement for control of unauthorized disclosure. Because the military security model provides a good starting point, we begin with a brief summary of computer security in the context of classified information control.

The top-level goal for the control of classified information is very simple: classified information must not be disclosed to unauthorized individuals. At first glance, it appears the correct mechanism to enforce this policy is a control over which individuals can read which data items. This mechanism, while certainly needed, is much too simplistic to solve the entire problem of unauthorized information release. In particular, enforcing this policy requires a mechanism to control writing of data as well as reading it. Because the control of writing data is superficially associated with ensuring integrity rather than preventing theft, and the classification policy concerns the control of theft, confusion has arisen about the fact that the military mechanism includes strong controls over who can write which data.

Informally, the line of reasoning that leads to this mechanism is as follows. To enforce this policy, the system must protect itself from the authorized user as well as the unauthorized user. There are a number of ways for the authorized user to declassify information. He can do so as a result of a mistake, as a deliberate illegal action, or because he invokes a program on his behalf that, without his knowledge, declassifies data as a malicious side effect of its execution.

This class of program, sometimes called a "Trojan Horse" program, has received much attention within the military. To understand how to control this class of problem in the computer, consider how a document can be declassified in a noncomputerized context. The simple technique involves copying the document, removing the classification labels from the document with a pair of scissors, and then making another copy that does not have the classification labels. This second copy, which physically appears to be unclassified, can then be carried past security guards who are responsible for controlling the theft of classified documents. Declassification occurs by copying.

To prevent this in a computer system, it is necessary to control the ability of an authorized user to copy a data item. In particular, once a computation has read a data item of a certain security level, the system must ensure that any data items written by that computation have a security label at least as restrictive as the label of the item previously read. It is this

mandatory check of the security level of all data items whenever they are written that enforces the high level security policy.

An important component of this mechanism is that checking the security level on all reads and writes is mandatory and enforced by the system, as opposed to being at the discretion of the individual user or application. In a typical time sharing system not intended for multilevel secure operation, the individual responsible for a piece of data determines who may read or write that data. Such discretionary controls are not sufficient to enforce the military security rules because, as suggested above, the authorized user (or programs running on his behalf) cannot be trusted to enforce the rules properly. The mandatory controls of the system constrain the individual user so that any action he takes is guaranteed to conform to the security policy. Most systems intended for military security provide traditional discretionary control in addition to the mandatory classification checking to support what is informally called "need to know." By this mechanism, it is possible for the user to further restrict the accessibility of his data, but it is not possible to increase the scope in a manner inconsistent with the classification levels.

In 1983, the U.S. Department of Defense produced the Orange Book, which attempts to organize and document mechanisms that should be found in a computer system designed to enforce the military security policies. This document stresses the importance of mandatory controls if effective enforcement of a policy is to be achieved within a system. To enforce the particular policy of the Orange Book, the mandatory controls relate to data labels and user access categories. Systems in division C have no requirement for mandatory controls, while systems in divisions A and B specifically have these mandatory maintenance and checking controls for labels and user rights. (Systems in Division A are distinguished from those in B, not by additional function, but by having been designed to permit formal verification of the security principles of the system.)

Several security systems used in the commercial environment, specifically RACF, ACF/2, and CA-TopSecret, were recently evaluated using the Orange Book criteria. The C ratings that these security packages received would indicated that they did not meet the mandatory requirements of the security model as described in the Orange Book. Yet, these packages are used commonly in industry and viewed as being rather effective in their meeting of industry requirements. This would suggest that industry views security requirements somewhat differently than the security policy described in the Orange Book. The next section of the paper begins a discussion of this industry view.

COMMERCIAL SECURITY POLICY FOR INTEGRITY

Clearly, control of confidential information is important in both the commercial and military environments. However, a major goal of commercial data processing, often the most important goal, is to ensure integrity of data to prevent fraud and errors. No user of the system, even if authorized, may be permitted to modify data items in such a way that assets or accounting records of the company are lost or corrupted. Some mechanisms in the system, such as user authentication, are an integral part of enforcing both the commercial and military policies. However, other mechanisms are very different.

The high-level mechanisms used to enforce commercial security policies related to data integrity were derived long before computer systems came into existence. Essentially, there are two mechanisms at the heart of fraud and error control: the well-formed transaction, and separation of duty among employees.

The concept of the well-formed transaction is that a user should not manipulate data arbitrarily, but only in constrained ways that preserve or ensure the integrity of the data. A very common mechanism in well-formed transactions is to record all data modifications in a log so that actions can be audited later. (Before the computer, bookkeepers were instructed to write in ink, and to make correcting entries rather than erase in case of error. In this way the books themselves, being write-only, became the log, and any evidence of erasure was indication of fraud.)

Perhaps the most formally structured example of well-formed transactions occurs in accounting systems, which model their transactions on the principles of double entry bookkeeping. Double entry bookkeeping ensures the internal consistency of the system's data items by requiring that any modification of the books comprises two parts, which account for or balance each other. For example, if a check is to be written (which implies an entry in the cash account) there must be a matching entry on the accounts payable account. If an entry is not performed properly, so that the parts do not match, this can

be detected by an independent test (balancing the books). It is thus possible to detect such frauds as the simple issuing of unauthorized checks.

The second mechanism to control fraud and error, separation of duty, attempts to ensure the external consistency of the data objects: the correspondence between the data object and the real world object it represents. Because computers do not normally have direct sensors to monitor the real world, computers cannot verify external consistency directly. Rather, the correspondence is ensured indirectly by separating all operations into several subparts and requiring that each subpart be executed by a different person. For example, the process of purchasing some item and paying for it might involve subparts: authorizing the purchase order, recording the arrival of the item, recording the arrival of the invoice, and authorizing payment. The last subpart, or step, should not be executed unless the previous three are properly done. If each step is performed by a different person, the external and internal representation should correspond unless some of these people conspire. If one person can execute all of these steps, then a simple form of fraud is possible, in which an order is placed and payment made to a fictitious company without any actual delivery of items. In this case, the books appear to balance; the error is in the correspondence between real and recorded inventory.

Perhaps the most basic separation of duty rule is that any person permitted to create or certify a well-formed transaction may not be permitted to execute it (at least against production data). This rule ensures that at least two people are required to cause a change in the set of well-formed transactions.

The separation of duty method is effective except in the case of collusion among employees. For this reason, a standard auditing disclaimer is that the system is certified correct under the assumption that there has been no collusion. While this might seem a risky assumption, the method has proved very effective in practical control of fraud. Separation of duty can be made very powerful by thoughtful application of the technique, such as random selection of the sets of people to perform some operation, so that any proposed collusion is only safe by chance. Separation of duty is thus a fundamental principle of commercial data integrity control.

Therefore, for a computer system to be used for commercial data processing, specific mechanisms are needed to enforce these two rules. To ensure that data items are manipulated only by means of well-formed transactions, it is first necessary to ensure that a data item can be manipulated only by a specific set of programs. These programs must be inspected for proper construction, and controls must be provided on the ability to install and modify these programs, so that their continued validity is ensured. To ensure separation of duties, each user must be permitted to use only certain sets of programs. The assignment of people to programs must again be inspected to ensure that the desired controls are actually met.

These integrity mechanisms differ in a number of important ways from the mandatory controls for military security as described in the Orange Book. First, with these integrity controls, a data item is not necessarily associated with a particular security level, but rather with a set of programs permitted to manipulate it. Second, a user is not given authority to read or write certain data items, but to execute certain programs on certain data items. The distinction between these two mechanisms is fundamental. With the Orange Book controls, a user is constrained by what data items he can read and write. If he is authorized to write a particular data item he may do so in any way he chooses. With commercial integrity controls, the user is constrained by what programs he can execute, and the manner in which he can read or write data items is implicit in the actions of those programs. Because of separation of duties, it will almost always be the case that a user, even though he is authorized to write a data item, can do so only by using some of the transactions defined for that data item. Other users, with different duties, will have access to different sets of transactions related to that data.

MANDATORY COMMERCIAL CONTROLS

The concept of mandatory control is central to the mechanisms for military security, but the term is not usually applied to commercial systems. That is, commercial systems have not reflected the idea that certain functions, central to the enforcement of policy, are designed as a fundamental characteristic of the system. However, it is important to understand that the mechanisms described in the previous section, in some respects, are mandatory controls. They are mandatory in that the user of the system should not, by any sequence of operations, be able to modify the list of programs permitted to manipulate a particular data item or to modify the

list of users permitted to execute a given program. If the individual user could do so, then there would be no control over the ability of an untrustworthy user to alter the system for fraudulent ends.

In the commercial integrity environment, the owner of an application and the general controls implemented by the data processing organization are responsible for ensuring that all programs are well-formed transactions. As in the military environment, there is usually a designated separate staff responsible for assuring that users can execute transactions only in such a way that the separation of duty rule is enforced. The system ensures that the user cannot circumvent these controls. This is a mandatory rather than a discretionary control.

The two mandatory controls, military and commercial, are very different mechanisms. They do not enforce the same policy. The military mandatory control enforces the correct setting of classification levels. The commercial mandatory control enforces the rules that implement the well-formed transaction and separation of duty model. When constructing a computer system to support these mechanisms, very different low-level tools are implemented.

An interesting example of these two sets of mechanisms can be found in the Multics operating system, marketed by Honeywell Information Systems and evaluated by the Department of Defense in Class B2 of its evaluation criteria. A certification in Division B implies that Multics has mandatory mechanisms to enforce security levels, and indeed those mechanisms were specifically implemented to make the system usable in a military multilevel secure environment [WHITMORE]. However, those mechanisms do not provide a sufficient basis for enforcing a commercial integrity model. In fact, Multics has an entirely different set of mechanisms, called protection rings, that were developed specifically for this purpose [SCHROEDER]. Protection rings provide a means for ensuring that data bases can be manipulated only by programs authorized to use them. Multics thus has two complete sets of security mechanisms, one oriented toward the military and designed specifically for multilevel operation, and the other designed for the commercial model of integrity.

The analogy between the two forms of mandatory control is not perfect. In the integrity control model, there must be more discretion left to the administrator of the system, because the determination of what constitutes proper separation of duty can be done only by a comparison with application-specific criteria. The separation of duty determination can be rather complex, because the decisions for all the transactions interact. This greater discretion means that there is also greater scope for error by the security officer or system owner, and that the system is less able to prevent the security officer, as opposed to the user, from misusing the system. To the system user, however, the behavior of the two mandatory controls is similar. The rules are seen as a fundamental part of the system, and may not be circumvented, only further restricted, by any other discretionary control that exists.

COMMERCIAL EVALUATION CRITERIA

As discussed earlier, RACF, ACF/2, and CA-TopSecret were all reviewed using the Department of Defense evaluation criteria described in the Orange Book. Under these criteria, these systems did not provide any mandatory controls. However, these systems, especially when executed in the context of a telecommunications monitor system such as CICS or IMS, constitute the closest approximation the commercial world has to the enforcement of a mandatory integrity policy. There is thus a strong need for a commercial equivalent of the military evaluation criteria to provide a means of categorizing systems that are useful for integrity control.

Extensive study is needed to develop a document with the depth of detail associated with the Department of Defense evaluation criteria. But, as a starting point, we propose the following criteria, which we compare to the fundamental computer security requirements from the "Introduction" to the Orange Book. First, the system must separately authenticate and identify every user, so that his actions can be controlled and audited. (This is similar to the Orange Book requirement for identification.) Second, the system must ensure that specified data items can be manipulated only by a restricted set of programs, and the data center controls must ensure that these programs meet the well-formed transaction rule. Third, the system must associate with each user a valid set of programs to be run, and the data center controls must ensure that these sets meet the separation of duty rule. Fourth, the system must maintain an auditing log that records every program executed and the name of the authorizing user. (This is superficially similar to the Orange Book requirement for accountability, but

the events to be audited are quite different.)

In addition to these criteria, the military and commercial environments share two requirements. First, the computer system must contain mechanisms to ensure that the system enforces its requirements. And second, the mechanisms in the system must be protected against tampering or unauthorized change. These two requirements, which ensure that the system actually does what it asserts it does, are clearly an integral part of any security policy. These are generally referred to as the "general" or "administrative" controls in a commercial data center.

A FORMAL MODEL OF INTEGRITY

In this section, we introduce a more formal model for data integrity within computer systems, and compare our work with other efforts in this area. We use as examples the specific integrity policies associated with accounting practices, but we believe our model is applicable to a wide range of integrity policies.

To begin, we must identify and label those data items within the system to which the integrity model must be applied. We call these "Constrained Data Items," or CDIs. The particular integrity policy desired is defined by two classes of procedures: Integrity Verification Procedures, or IVPs, and Transformation Procedures, or TPs. The purpose of an IVP is to confirm that all of the CDIs in the system conform to the integrity specification at the time the IVP is executed. In the accounting example, this corresponds to the audit function, in which the books are balanced and reconciled to the external environment. The TP corresponds to our concept of the well-formed transaction. The purpose of the TPs is to change the set of CDIs from one valid state to another. In the accounting example, a TP would correspond to a double entry transaction.

To maintain the integrity of the CDIs, the system must ensure that only a TP can manipulate the CDIs. It is this constraint that motivated the term Constrained Data Item. Given this constraint, we can argue that, at any given time, the CDIs meet the integrity requirements. (We call this condition a "valid state.") We can assume that at some time in the past the system was in a valid state, because an IVP was executed to verify this. Reasoning forward from this point, we can examine the sequence of TPs that have been executed. For the first TP executed, we can assert that it left the system in a valid state as follows. By definition it will take the CDIs into a valid state if they were in a valid state before execution of the TP. But this precondition was ensured by execution of the IVP. For each TP in turn, we can repeat this necessary step to ensure that, at any point after a sequence of TPs, the system is still valid. This proof method resembles the mathematical method of induction, and is valid provided the system ensures that only TPs can manipulate the CDIs.[1]

While the system can ensure that only TPs manipulate CDIs, it cannot ensure that the TP performs a well-formed transformation. The validity of a TP (or an IVP) can be determined only by certifying it with respect to a specific integrity policy. In the case of the bookkeeping example, each TP would be certified to implement transactions that lead to properly segregated double entry accounting. The certification function is usually a manual operation, although some automated aids may be available.

Integrity assurance is thus a two-part process: certification, which is done by the security officer, system owner, and system custodian with respect to an integrity policy; and enforcement, which is done by the system. Our model to this point can be summarized in the following three rules:

C1: (Certification) All IVPs must properly ensure that all CDIs are in a valid state at the time the IVP is run.

C2: All TPs must be certified to be valid. That is, they must take a CDI to a valid final state, given that it is in a valid state to begin with. For each TP, and each set of CDIs that it may manipulate, the security officer must specify a "relation," which defines that execution. A relation is thus

---

[1]There is an additional detail which the system must enforce, which is to ensure that TPs are executed serially, rather than several at once. During the mid-point of the execution of a TP, there is no requirement that the system be in a valid state. If another TP begins execution at this point, there is no assurance that the final state will be valid. To address this problem, most modern data base systems have mechanisms to ensure that TPs appear to have executed in a strictly serial fashion, even if they were actually executed concurrently for efficiency reasons.

of the form: (TPi, (CDIa, CDIb, CDIc, . . .)), where the list of CDIs defines a particular set of arguments for which the TP has been certified.

E1: (Enforcement) The system must maintain the list of relations specified in rule C2, and must ensure that the only manipulation of any CDI is by a TP, where the TP is operating on the CDI as specified in some relation.

The above rules provide the basic framework to ensure internal consistency of the CDIs. To provide a mechanism for external consistency, the separation of duty mechanism, we need additional rules to control which persons can execute which programs on specified CDIs:

E2: The system must maintain a list of relations of the form: (UserID, TPi, (CDIa, CDIb, CDIc, ...)), which relates a user, a TP, and the data objects that TP may reference on behalf of that user. It must ensure that only executions described in one of the relations are performed.

C3: The list of relations in E2 must be certified to meet the separation of duty requirement.

Formally, the relations specified for rule E2 are more powerful than those of rule E1, so E1 is unnecessary. However, for both philosophical and practical reasons, it is helpful to have both sorts of relations. Philosophically, keeping E1 and E2 separate helps to indicate that there are two basic problems to be solved: internal and external consistency. As a practical matter, the existence of both forms together permits complex relations to be expressed with shorter lists, by use of identifiers within the relations that use "wild card" characters to match classes of TPs or CDIs.

The above relation made use of UserID, an identifier for a user of the system. This implies the need for a rule to define these:

E3: The system must authenticate the identity of each user attempting to execute a TP.

Rule E3 is relevant to both commercial and military systems. However, those two classes of systems use the identity of the user to enforce very different policies. The relevant policy in the military context, as described in the Orange Book, is based on level and category of clearance, while the commercial policy is likely to be based on separation of responsibility among two or more users.

There may be other restrictions on the validity of a TP. In each case, this restriction will be manifested as a certification rule and enforcement rule. For example, if a TP is valid only during certain hours of the day, then the system must provide a trustworthy clock (an enforcement rule) and the TP must be certified to read the clock properly.

Almost all integrity enforcement systems require that all TP execution be logged to provide an audit trail. However, no special enforcement rule is needed to implement this facility; the log can be modeled as another CDI, with an associated TP that only appends to the existing CDI value. The only rule required is:

C4: All TPs must be certified to write to an append-only CDI (the log) all information necessary to permit the nature of the operation to be reconstructed.

There is only one more critical component to this integrity model. Not all data is constrained data. In addition to CDIs, most systems contain data items not covered by the integrity policy that may be manipulated arbitrarily, subject only to discretionary controls. These Unconstrained Data Items, or UDIs, are relevant because they represent the way new information is fed into the system. For example, information typed by a user at the keyboard is a UDI; it may have been entered or modified arbitrarily. To deal with this class of data, it is necessary to recognize that certain TPs may take UDIs as input values, and may modify or create CDIs based on this information. This implies a certification rule:

C5: Any TP that takes a UDI as an input value must be certified to perform only valid transformations, or else no transformations, for any possible value of the UDI. The transformation should take the input from a UDI to a CDI, or the UDI is rejected. Typically, this is an edit program.

For this model be effective, the various certification rules must not be bypassed. For example, if a user can create and run a new TP without having it certified, the system cannot meet its goals. For this reason, the system must

ensure certain additional constraints. Most obviously:

E4: Only the agent permitted to certify entities may change the list of such entities associated with other entities: specifically, the associated with a TP. An agent that can certify an entity may not have any execute rights with respect to that entity.

This last rule makes this integrity enforcement mechanism mandatory rather than discretionary. For this structure to work overall, the ability to change permission lists must be coupled to the ability to certify, and not to some other ability, such as the ability to execute a TP. This coupling is the critical feature that ensures that the certification rules govern what actually happens when the system is run.

Together, these nine rules define a system that enforces a consistent integrity policy. The rules are summarized in Figure 1, which shows the way the rules control the system operation. The figure shows a TP that takes certain CDIs as input and produces new versions of certain CDIs as output. These two sets of CDIs represent two successive valid states of the system. The figure also shows an IVP reading the collected CDIs in the system in order to verify the CDIs' validity. Associated with each part of the system is the rule (or rules) that governs it to ensure integrity.

Central to this model is the idea that there are two classes of rules: enforcement rules and certification rules. Enforcement rules correspond to the application-independent security functions, while certification rules permit the application-specific integrity definitions to be incorporated into the model. It is desirable to minimize certification rules, because the certification process is complex, prone to error, and must be repeated after each program change. In extending this model, therefore, an important research goal must be to shift as much of the security burden as possible from certification to enforcement.

For example, a common integrity constraint is that TPs are to be executed in a certain order. In the model (and in most systems of today), this idea can be captured only by storing control information in some CDI, and executing explicit program steps in each TP to test this information. The result of this style is that the desired policy is hidden within the program, rather than being stated as an explicit rule that the system can then enforce.

Other examples exist. Separation of duty might be enforced by analysis of sets of accessible CDIs for each user. We believe that further research on specific aspects of integrity policy would lead to a new generation of tools for integrity control.

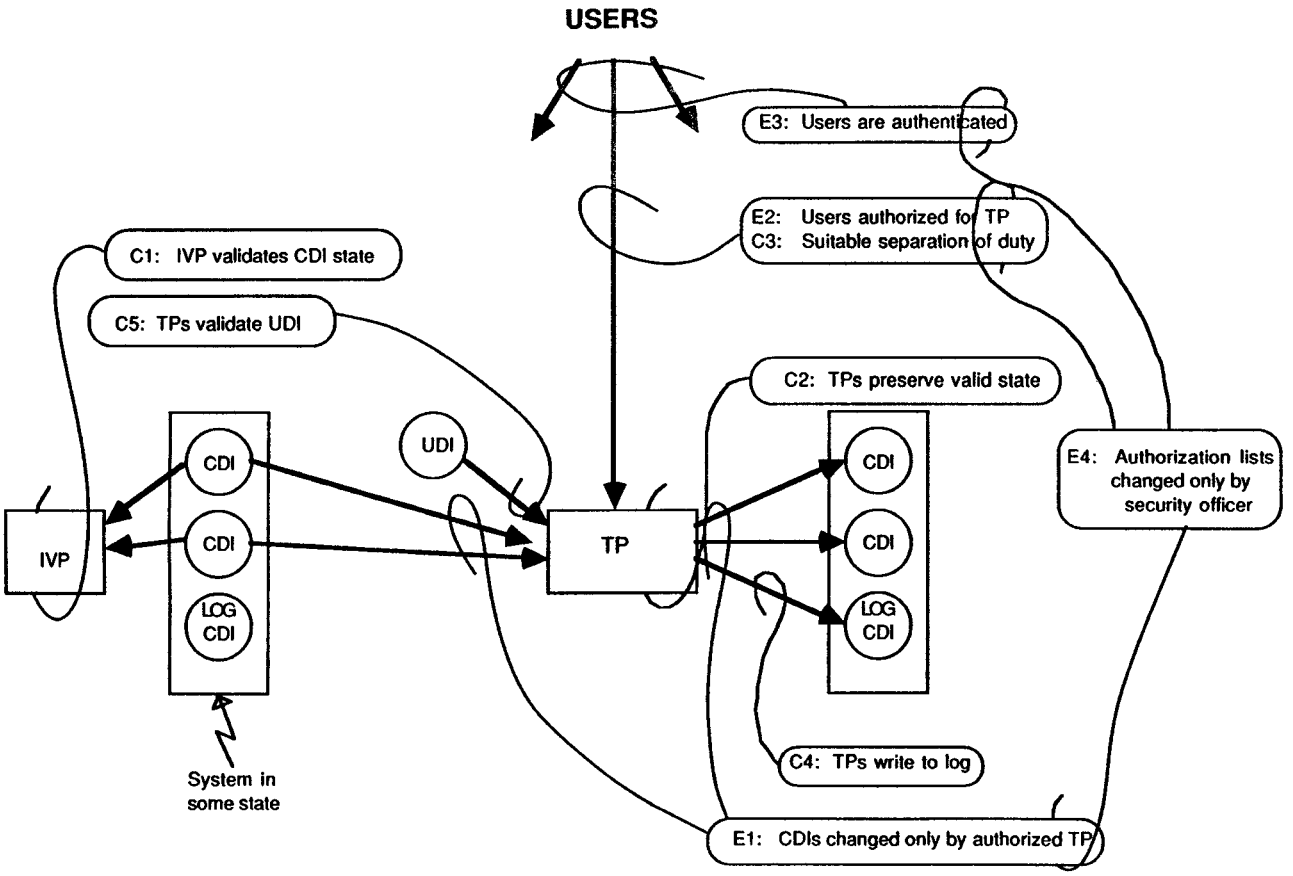
OTHER MODELS OF INTEGRITY

Other attempts to model integrity have tried to follow more closely the structure for data security defined by Bell and LaPadula [BELL], the formal basis of the military security mechanisms. Biba [BIBA] defined an integrity model that is the inverse of the Bell and LaPadula model. His model states that data items exist at different levels of integrity, and that the system should prevent lower level data from contaminating higher level data. In particular, once a program reads lower level data, the system prevents that program from writing to (and thus contaminating) higher level data.

Our model has two levels of integrity: the lower level UDIs and the higher level CDIs. CDIs would be considered higher level because they can be verified using an IVP. In Biba's model, any conversion of a UDI to a CDI could be done only by a security officer or trusted process. This restriction is clearly unrealistic; data input is the most common system function, and should not be done by a mechanism essentially outside the security model. Our model permits the security officer to certify the method for integrity upgrade (in our terms, those TPs that take UDIs as input values), and thus recognizes the fundamental role of the TP (i.e., trusted process) in our model. More generally, Biba's model lacks any equivalent of rule E1 (CDIs changed only by authorized TP), and thus cannot provide the specific idea of constrained data.

Another attempt to describe integrity using the Bell and LaPadula model is Lipner [LIPNER]. He recognizes that the category facility of this model can be used to distinguish the general user from the systems programmer or the security officer. Lipner also recognizes that data should be manipulated only by certified (production) programs. In attempting to express this in terms of the lattice model, he is constrained to attach lists of users to programs and data separately, rather than attaching a list of programs to a data item. His model thus has no way to express our rule E1. By combining a lattice security model with the Biba integrity model, he more closely approximates the desired model,

Figure 1: Summary of System Integrity Rules



but still cannot effectively express the idea that data may be manipulated only by specified programs (rule E1).

Our integrity model is less related to the Bell and LaPadula model than it is to the models constructed in support of security certification of systems themselves. The iterative process we use to argue that TPs preserve integrity, which starts with a known valid state and then validates incremental modifications, is also the methodology often used to verify that a system, while executing, continues to meet its requirements for enforcing security. In this comparison, our CDIs would correspond to the data structures of the system, and the TPs to the system code. This comparison suggests that the certification tools developed for system security certification may be relevant for the certifications that must be performed on this model.

For example, if an Orange Book for industry were created, it also might have rating levels. Existing systems such as ACF/2, RACF, and CA-TopSecret would certainly be found wanting in comparison to the model. This model would suggest that, to receive higher ratings, these security systems must provide: better facilities for end-user authentication; segregation of duties within the security officer functions, such as the ability to segregate the person who adds and deletes users from those who write a user's rules, and restriction of the security function from user passwords; and the need to provide much better rule capabilities to govern the execution of programs and transactions.

The commercial sector would be very interested in a model that would lead to and measure these kinds of changes. Further, for the commercial world, these changes would be much more valuable than to take existing operating systems and security packages to B or A levels as defined in the Orange Book.

CONCLUSION

With the publication of the Orange Book, a great deal of public and governmental interest has focused on the evaluation of computer systems for security. However, it has been difficult for the commercial sector to evaluate the relevance of the Orange Book criteria, because there is no clear articulation of the goals of commercial security. This paper has attempted to identify and describe one such goal, information integrity, a goal that is central to much of commercial data processing.

In using the words commercial and military in describing these models, we do not mean to imply that the commercial world has no use for control of disclosure, or that the military is not concerned with integrity. Indeed, much data processing within the military exactly matches commercial practices. However, taking the Orange Book as the most organized articulation of military concerns, there is a clear difference in priority between the two sectors. For the core of traditional commercial data processing, preservation of integrity is the central and critical goal.

This difference in priority has impeded the introduction of Orange Book mechanisms into the commercial sector. If the Orange Book mechanisms could enforce commercial integrity policies as well as those for military information control, the difference in priority would not matter, because the same system could be used for both. Regrettably, this paper argues there is not an effective overlap between the mechanisms needed for the two. The lattice model of Bell and LaPadula cannot directly express the idea that manipulation of data must be restricted to well-formed transformations, and that separation of duty must be based on control of subjects to these transformations.

The evaluation of RACF, ACF/2, and CA-TopSecret against the Orange Book criteria has made clear to the commercial sector that many of these criteria are not central to the security concerns in the commercial world. What is needed is a new set of criteria that would be more revealing with respect to integrity enforcement. This paper offers a first cut at such a set of criteria. We hope that we can stimulate further effort to refine and formalize an integrity model, with the eventual goal of providing better security systems and tools in the commercial sector.

There is no reason to believe that this effort would be irrelevant to military concerns. Indeed, incorporation of some form of integrity controls into the Orange Book might lead to systems that better meet the needs of both groups.

ACKNOWLEDGMENTS

The authors would like to thank Robert G. Andersen, Frank S. Smith, III, and Ralph S. Poore (Ernst & Whinney, Information Security Services) for their assistance in preparing this paper. We also thank Steve Lipner (Digital Equipment Corporation) and the referees of the paper for their very helpful comments.

REFERENCES

[DoD] Department of Defense Trusted Computer System Evaluation Criteria, CSC-STD-011-83, Department of Defense Computer Security Center, Fort Meade, MD, August 1983.

[Bell] Bell, D. E. and L. J. LaPadula, "Secure Computer Systems," ESD-TR-73-278 (Vol I-III) (also Mitre TR-2547), Mitre Corporation, Bedford, MA, April 1974.

[Biba] Biba, K. J., "Integrity Considerations for Secure Computer Systems," Mitre TR-3153, Mitre Corporation, Bedford, MA, April 1977.

[Lipner] Lipner, S. B., "Non-Discretionary Controls for Commercial Applications," Proceedings of the 1982 IEEE Symposium on Security and Privacy, Oakland, CA, April 1982.

[Schroeder] Schroeder, M. D. and J. H. Saltzer, "A Hardware Architecture for Implementing Protection Rings," Comm ACM, Vol 15, 3 March 1972.

[Whitmore] Whitmore, J. C. et al., "Design for Multics Security Enhancements," ESD-TR-74-176, Honeywell Information Systems, 1974.