External Consistency in a Networking Environment

Leonard J. LaPadula and James G. Williams

The MITRE Corporation

Abstract Paradigms for information integrity in computer systems have focused in the past on modeling a computer's internal mechanisms. Biba integrity and recent attempts to "latticize" any computer-based policy are examples. Traditional paradigms for enterprise integrity focus on business procedures without regard to computer systems, double-entry bookkeeping being one example. Taking cues from the 1987 work of Clark and Wilson, we have modeled the interface between an enterprise and its computer system. In earlier work, we described a rigorous external-consistency model that defines user and computer responsibilities and the relationships between them. In this white paper we raise issues for a new paradigm — to extend our external consistency model to the interface between an enterprise and a distributed automated system.

INTRODUCTION

In this paper we pose a problem for discussion at the New Security Paradigms II workshop. We have a model of external consistency for a trusted computer system [1] that looks promising, but it is oriented toward a single, multi-user computer system. The problem is how to apply it to a distributed processing environment.

At last year's workshop the attendees identified numerous shifts that seemed desirable. The three shifts most relevant to our modeling of external consistency are:

- From emphasis on technical security in relative ignorance of usage, operational needs, and environment to cognizance and treatment of all aspects of the usage of computers in an enterprise
- From a narrow view of security modeling, focusing on internal requirements and rules of operation, to modeling at higher levels of abstraction, including enterprise modeling
- From the current reference monitor concept to a more general version to accommodate the need for user-system mediation, system-dependent policies, multiple policies, and integrity and availability requirements

Over the past year we have developed a model of external consistency for automated systems. The model reflects these shifts in thinking and emphasis. However, our orientation in developing external consistency requirements focuses on the environment and services normally associated with a single, multi-user computer system. We believe that the model has wider applicability to many configurations of automated systems. We also believe, though, that in looking explicitly at distributed environments we will inevitably discover additionally needed requirements and interesting corollaries to our work. In this paper we will give an informal description of our external consistency model and then frame a problem for discussion by the attendees of the workshop.

A SUMMARY OF THE EXTERNAL-CONSISTENCY MODEL

Perspective

Last year, at the debut of the New Security Paradigms Workshop, we introduced a taxonomy of stages in defining requirements for a trusted computer system [2]. This taxonomy also provides useful background for our external-consistency work. After the discussions at the workshop in 1992, we expanded the taxonomy from its original five stages to the following seven stages.

Table 1. The First Four Stages of the Requirements Taxonomy

STAGE OF ELABORATION	DESCRIPTION
1 -- Enterprise Description	An *enterprise description* produces a model of an enterprise, describing the activities, responsibilities, and methods that realize the goals of the enterprise, such as a description of roles, responsibilities, interactions with customers, and data flow at enterprise sites. The enterprise description gives needed background for stating the trust objectives for the enterprise.
2 -- Trust Objectives	A *trust objective* identifies a goal that counters information-related misfortunes in an enterprise, an important component of which is a computing system.
3 -- External-Interface Requirements Model	An *external-interface requirements model* describes the behaviors of the computing system, its users, and other entities in the system's environment in such a way as to allocate responsibilities for achieving the identified trust objectives, thereby showing how the system supports the identified trust objectives.
4 -- Internal Requirements Model	An *internal requirements model* describes, in an abstract manner, how the system responsibilities given in the external model are met within the system.

- Enterprise Description
- Trust Objectives
- External-Interface Requirements Model
- Internal Requirements Model
- Rules of Operation
- Functional Design
- Hardware and Software Description

The first four of these stages are most relevant to the work we have done; they are described in table 1.

Our work on external-consistency models has focused on stages 2 and 3. Work like Dobson's [3] focuses on stage 1. One could profitably use Dobson's approach to describe the environment for application of an automated system satisfying our external-consistency model. In developing our model, we have had in mind an enterprise of the general kind described by Clark and Wilson [4].

Clark and Wilson argued that

- 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.
- Some separate mechanisms are required for enforcement of these policies, disjoint from those of the Orange Book.

We agree with the main tenets of their arguments. However, while they and others have put their attention on the foundational characteristics of mechanisms, we have examined fundamental aspects of an integrity policy itself. The aspect of integrity we are particularly interested in is external consistency.

The External-Consistency Objective

In the context of computing systems, *external consistency* is the ability of a computing system to give correct information about its external environment. Some typical examples where external consistency is important are as follows.

- An inventory program reports that a warehouse contains given levels of various supplies: the named supplies should actually be at the warehouse in the amounts claimed.

- A bank statement produced by an automated system lists a particular balance for an account: the balance should be correct at the time it is issued.
- A computing system issues a "sell" order for one million shares of stock: the order should reflect the intent of its controlling organization.

Key to modeling external consistency is the fact that computers produce output that users interpret as statements about real-world entities. We envision a situation in which users exchange assertions, requests, and questions with an automated information system. In our modeling, we require each input and output to be cast in terms of a stable proposition - that is, a proposition whose truth is independent of the time the assertion is made and whose precision is specified within the proposition itself. In the informal language of this paper, we simply refer to inputs and outputs, understanding that careful definition of these terms is needed for formal modeling.

The *external-consistency objective* is that each output assertion or request be real-world correct. For an assertion, correctness means correspondence to reality - each assertion received from the system is a true description of reality. This objective applies to all assertions made by the system, including account status reports, financial transactions, purported facts about the behavior of programs, and so forth. For a request, correctness means legitimacy according to some preassigned criteria - each request received from the system is legitimate according to some enterprise-specific or application-specific set of criteria. We assume that each enterprise or application provides criteria for judging the legitimacy of requests associated with that enterprise or application.

Basic Responsibilities

The first step in elaborating external-consistency objectives is to allocate responsibility between a system and its users, identified as stage 3 in our taxonomy. We specify an allocation of responsibilities that allows the vendor to actively share responsibility for achieving the external-consistency objective.

Automated System Responsibilities

In devising requirements for the automated system, we want to determine what the vendor-supplied hardware/software configuration should do to support the external-consistency objective, on the assumption that the system is installed properly and then not tampered with or inappropriately modified. There are some output assertions that a properly installed system, and its vendor, might accept full responsibility for. An example is assertions based on reading the system clock, a trusted input device that directly observes the passage of time.

The correctness of most outputs, however, depends on the correctness of previous relevant inputs. Consider the stock-market sell order. Besides issuing the order, the computer system might assert that the sell order was approved by an authorized user other than the one initiating the sell order. We expect such an assertion to be true. However, the computer system cannot force the assertion to be true; it can, at best, assert that it has received a sell request and an approval by an authenticated user who seemed to be different from the one making the request.

We want the computer's sell order to be a correct representation of the will of the enterprise it works for. In terms of the external-consistency objective, the sell order should be a true description of reality. What the vendor and the computer system can do is warrant the sell order - they can guarantee that if the user-supplied information on which the sell order is based is correct, then so is the sell order. For reasons of audit and to serve other, related enterprise activities such as error correction, we would like the computer system to maintain an accurate record of the particular information that supports the sell order - we call this set of information an input/output (I/O) *basis*. In effect, the vendor supplies a limited warranty, where the limitations for a given warranted output are given by its I/O basis.

For reasons of efficiency, we assume that not all system outputs must meet the external-consistency objective. Those that should can be marked as warranted by the system. Previous outputs may be included in an I/O basis to allow a more natural correspondence with typical business practices. For example, the basis for

this month's bank statement includes the closing balance from last month's statement, which is appropriate, if it wasn't contested.

User Responsibilities

External consistency depends on correct inputs. The fundamental responsibility of the users is to provide correct inputs to the automated system. In the simplest of models one would require that all assertions and requests made to the system by its users be correct. This requirement will not, in general, be met in the real world. Thus, for the sake of utility, we are led to consider refinements.

Refinements

Consider situations having a mix of naive and experienced users, in which experienced users are able to avoid mistakes and catch errors made by others. Central to this situation is the fact that the automated system will be receiving both correct and incorrect inputs. Thus, the automated system needs procedures and mechanisms by which it can distinguish warrantable outputs, those based on correct inputs, from outputs that may have been contaminated by incorrect inputs. We begin with some examples and observations.

The software that implements a bank's teller functions might employ an underlying database of transaction constraints, managed by a designated administrator. This constraints-database might rule out negative checking balances and might impose minimum balances for certain types of accounts. The teller software then simply disallows teller updates not consistent with the constraints-database. Thus, we can distinguish inputs of the "experienced" user from those of the "naive" user. The automated system considers the rules of the constraints-database's administrator correct, and subsequent attempted updates of a contrary nature incorrect. Also, the bank may require tellers to get approval from bank officers for certain classes of transactions. An input in these cases is considered correct by the teller software when it is appropriately corroborated by a different authorized user. Thus, we require that "experienced" users certify to the automated system that certain kinds of inputs are correct. The automated system should then treat all other inputs as potentially incorrect because, for example, they are given by "inexperienced" users. The automated system must then internally keep track of data derived from or based on correct inputs so that it can tell which of its outputs it can warrant.

An authorized administrator can use a *correctness-ensuring profile* to certify a type of input as being correct. The profile consists of input-event attributes such as type of data, the input device or communications path used, whether the input has been corroborated by previous inputs with specified input attributes, and, for user inputs, user identity, user role, and explicit evidence of correctness provided by the user. Given a collection of correctness-ensuring profiles, the automated system can tell users which outputs are based on correct inputs and are, therefore, correct. It can also refuse to use inputs when they are contraindicated by correctness-ensuring profiles.

The system also can run user-supplied or built-in integrity validators to catch incorrect inputs. *Integrity validators* are conceptually similar to those transformation procedures of Clark and Wilson [4] that are used to transform inputs into constrained data items. Naturally, the user-supplied integrity validators must be certified to the system as being correct inputs.

Thus, the automated system accepts an input as being correct if it can warrantably claim that its input-event profile is correctness-ensuring and that the input has been approved by all relevant, certified integrity-validators. We refer to such inputs as being "certified."

These refinements lead to the following definition of responsibilities for the automated system and its users.

Automated System Responsibilities

The automated system must keep account of which outputs are warrantable and must identify such outputs as warranted. The conditions that pertain for *warranted outputs* are as follows.

- Every output marked as warranted is warrantable in the sense that it has a credible I/O basis and it is correct given that each item of its I/O basis is correct.

- An I/O basis is *credible* if and only if its inputs are certified and its outputs are warrantable - that is, it is a warrantable basis whose inputs are certified.
- An input is *certified* if and only if (a) it is a direct observation by the system, or (b) it has been approved by all relevant, certified integrity validators and the automated system can warrantably claim that its input-event profile is correctness-ensuring.

User Responsibilities

The fundamental user responsibility can now be expressed as an input-correctness responsibility: All certified inputs to the system must be correct.

For the automated system to make use of certified inputs - that is, to realize utility for the enterprise - the administrators or "experienced users" of the system must carry out their roles in defining correctness-ensuring profiles, installing integrity validators, and so forth.

Modeling More Capable Systems

We have pursued the formal modeling of external consistency beyond the framework reported in [1], to include error correction and retraction. We give a summary here of the desired objectives and corresponding user and system requirements, since they are of interest for the problem we pose later.

In the real world even certified inputs may be incorrect. Hence, we must settle for a less absolute form of the external-consistency objective. The best one can hope for is to discover such errors after the fact and then limit their consequences. We require, at a minimum, that failures of the objective be traceable. That is, if a given output is incorrect, we should be able to identify erroneous inputs or system failings that caused the incorrect output. If the system is functioning properly, an output error can only be the result of some erroneous input. This gives rise to an *accountability objective for external consistency*: If a warranted output is incorrect, then qualified error investigators will determine the cause of that error. That is, they will learn whether that error is due to a system malfunction and, if not, what incorrect inputs supported the erroneous output. We imagine, for example, that authorized administrators assign reliable error reporters and error investigators who inform the automated system of previous incorrect inputs through certified inputs. When an output error is reported to the automated system, the automated system must cooperate by collecting and providing relevant historical data such as the I/O bases of outputs involved in the error.

When a bank deposits a paycheck in the wrong account, several kinds of actions are necessary to address the error fully. The original deposit must be voided and replaced with the correct deposit. It is also necessary to know how the error has propagated, so that erroneous outputs to creditors and credit bureaus can be rescinded. Corrective input to the bank's automated system should be treated as more credible than a previous certified but incorrect input, and later outputs should be regarded as more credible than the bad-credit reports they rescind, even though those reports were warranted. Considerations like these give rise to an *error-suppression objective*: Errors fail to propagate once they are recognized.

Restoration of external consistency after an error depends on the availability of corrective information. To model this situation we develop the notion of a supplanting relation among inputs. Input B supplants input A only if an authorized administrator has provided an input to the automated system certifying that input B is more credible than input A. This notion enables the automated system to issue retractions. The *error-retraction objective* is: Whenever a certified input is found to be incorrect by an authorized error investigator, each retractable output which that input supports is retracted by some later, more credible output to the same recipients, and possibly others as well.

THE PROBLEM

How would you apply our external consistency model to a distributed systems environment? For discussing this question we posit a network of secure wallets and then suggest some of the issues to be addressed.

The Network

The "secure-wallet network" is a collection of secure electronic wallets and one or more bank servers. An electronic wallet is essentially a wallet-size, special-purpose computer that enables its owner to transfer money. A bank server is fundamentally a third party to transactions to provide an official record, support audit, enforce nonrepudiation, and so forth.

In our imaginary network, electronic wallets can interface to each other directly, either port-to-port or via common-carrier phone lines. They can also interface to bank servers by the same methods. Although many uses can be envisioned for such a network, we restrict our attention to "cash" transfers, essentially the operation and management of electronic checking accounts. To motivate subsequent discussion of issues, we describe the following scenario involving Person-One, Person-Two, Bar-Wallet, Bank Server-One, and Bank Server-Two.

- Person-One and Person-Two go out to a Tavern for friendly drinks and a game of darts. To make the dart games more interesting, the following conditions are agreed to: loser for the evening pays the winner $60, winner pays for all the evening's drinks. (Note that this setup has positive feedback - losing player tends to drink more, thereby lessening chances of winning, and winning player tends to drink less to reduce the bar bill. We do not claim "political correctness.")
- At the end of the evening, Person-One is the loser of the dart game. Person One and Person Two put their electronic wallets "port-to-port" and Person-One instructs Wallet-One to transfer $60 to Wallet-Two. Person-Two then puts Wallet-Two "port-to-port" with the Bar-Wallet and authorizes transfer of $76 (that includes tax and gratuity) to the Tavern.
- Next morning, Person-Two attaches Wallet-Two to Bank Server-Two via phone line. Wallet-Two reports the previous day's transactions and receives acknowledgment of those transactions. Similarly the Bar-Wallet "plugs in" to Bank Server-One and, among other transactions, gets confirmation of payment of the $76 by Person-Two. Later that same morning, Person-One groggily dials up Bank Server-One, attaches Wallet-One to the telephone interface, and instructs Wallet-One to report the previous day's transactions.

The Issues

There are plenty of issues one could address. Since we wish to support the external consistency objective we have discussed, we are interested in general questions of the following kind.

- How should the automated system requirements be allocated across the components of the network?
- What fine tuning or tailoring of the user requirements is necessary?
- What additional automated system or user requirements arise?

More specific questions can be asked; some are:

- Is the owner of an electronic wallet, in addition to being a user, to be considered an authorized administrator for some purposes?
- Are integrity validators useful in both the electronic wallet and the bank server?
- If electronic wallets and bank servers are provided by different vendors, how do the vendors cooperatively share responsibility for achieving the external consistency objective?
- Must electronic wallets maintain I/O bases? If so, for how long? Does it suffice to maintain them, for example, only until the next reporting of transactions to a bank server?
- Can the notion of endorsed inputs be applied here? For example, similarly to use of a cashier's check, it might be possible to have a three-way transaction in which Bank Server-One "endorses" the transfer of a large sum from Wallet-One to Wallet-Two.
- In the case of nonendorsed transfers, nonrepudiation becomes important. How can nonrepudiation be ensured?
- Which system elements are most crucial to access mediation and what do they do?

REFERENCES

[1] Williams, J. G. and L. J. LaPadula, June 1993, "Automated Support for External Consistency", *Proceedings of the Sixth IEEE Workshop on Computer Security Foundations*, Franconia, NH.

[2] LaPadula, L. J., September, 1992, "Prospect on Security Paradigms", *Proceedings of the ACM Workshop on New Security Paradigms*, Little Compton, RI.

[3] Dobson, J. E. and J. A. McDermid, 1989, "Security Models and Enterprise Models", in Database Security: Status and Prospects II, ed. C. E. Landwehr, pp. 1-39, Elsevier Science Publishers, Amsterdam.

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