Confidentiality in a Replicated Architecture Trusted Database System: A Formal Model[1]

Oliver Costich

Center for Secure Information Systems
George Mason University
4400 University Drive
Fairfax, Virginia 22030

John McLean
John McDermott

Center for High Assurance
Computer Systems
Naval Research Laboratory
Washington, D.C. 20375

Abstract

Unlike previous approaches to developing a trusted database system, the replicated architecture approach provides access control at a high level of assurance through replication of data and operations. We present a model of the SINTRA replicated architecture trusted database system which shows how the logical (users') view of the system and its security policy is translated into the physical structure and operations of the SINTRA system. We formalize the intended security policy for replicated architecture and demonstrate that a high level of assurance can be obtained solely from replication with virtually no change to the structure of the underlying database systems or the security kernel.

1. Introduction

The replicated architecture for trusted databases was proposed in [5]. The underlying motivation for this approach was improved performance and the ability to get access control easily. It also appears that commercially available databases can be used for much of the system, providing economies of cost and development time. The SINTRA project at The Naval Research Laboratory has developed a prototype trusted database system based on this architecture.

The purpose of this paper is to demonstrate that the confidentiality (as opposed to integrity) component of a security policy is enforced as a function of the architecture alone, without any loss of database functionality. That is, confidentiality is obtained in the replicated architecture without adding any trusted mechanisms to the database system itself, relying solely on those in the the front end component, as described below.

While it is not difficult to see that confidentiality is enforced "free" in this system, there has been some confusion regarding the replicated architecture model which we hope to resolve in this paper. To this end, we will present a formal security policy model for a replicated architecture trusted database. From the model it will be apparent that the confidentiality (as opposed to integrity) component of security policy is enforced as a function of the architecture alone. That is, virtually no argument is necessary that the system provides confidentiality. Integrity policy is not so apparent (and may be arguable from some points of view), and will be discussed as well.

We should note that the usefulness of the replicated architecture approach is not limited to database systems, but can also be used in many other trusted system applications, and that the level of assurance of

---

[1] Appears in *Proceedings of the Computer Security Foundations Workshop VII*, pages 60-65, IEEE Computer Society, 1994.

the resulting system will be the same as that of the front end component.

We present the logical (user) view of a trusted database followed by a discussion of the SINTRA architecture. We proceed to the development of the formal model and then discuss the ramifications. Finally, we look at other modeling considerations, and the application of this architecture to trusted applications other than databases.

2. The Logical View of a Trusted Database

The logical view of a system is the view of the system as it appears to the user. To the user, a database consists of a collection of *data items*. They access these data items via transactions built as sequences of *read* and *write* operations on the data items. Operations inherit a security level from the log-on clearance of the user on whose behalf they are being executed. Data items have assigned *security levels* and these security levels form a lattice.

From this point of view, we would like to enforce a security policy similar to the Bell-LaPadula criteria [1].

1) Simple Security Property: A *read* operation can access a data item if and only if its security level dominates that of the data item.

2) Restricted ⋆-Property: A *write* operation can access a data item if and only if its security level equals that of the data item.

Implicit in this criteria is that the subjects of the logical system are the operations and the objects, data items.

When databases are actually designed, multiple copies or replicas of data items are frequently used to improve performance by promoting concurrency of transaction execution. This creates the problem of reconciling the logical, single-copy view of the database with the physical view of the system which deals with copies because the management of the copies is transparent to the users. In the case at hand, we will adopt replication of data items, which creates a similar situation, not only to improve performance but also to obtain certain security features more easily.

3. The SINTRA Architecture and Security Policy

The SINTRA replicated architecture utilizes a physically separate backend database for each security level in the security lattice. Each backend database can be thought of as container of the appropriate security level. In addition, access to the backend databases is mediated by a trusted front-end system (TFE).

The TFE performs the usual authorization and identification functions, and also insures that a user who is logged on at a particular security level has access to only the single backend database of the corresponding security level. Other functions handled by the TFE may include those which require separation of entities to obtain database functionality, e.g., queues for concurrency control.

Not all of the TFE need be trusted, but the trusted portion must have a level of assurance at least as high as that required by the whole database. Thus for a high assurance database, which is our concern here, the trusted portion of the TFE (usually an operating system) will have its own security policy model. The issue for us then becomes what must be added to it to obtain a model for the rest of the database.

A data item of the logical database is represented in the backend database by placing a replica of that data item in each backend database whose security level dominates the logical security level of the data item. In other words, if data item x has logical security level a, then a copy of x appears in every backend database whose security level dominates a. Objects are these data items, which, of course, now include all replicas of the logical data items. These replicas will be referred to as *physical* data items to distinguish them from the *logical* ones.

Similarly, subjects of SINTRA are of two types. The first are the *logical read* and *write* operations of user-submitted transactions, and their security level is determined by the clearance of the user who submitted them. Second, each logical write operation of a user submitted transaction must also be executed at each higher security level backend database in order to maintain consistency in the values of the replicas. The physical write operations which update the replicas of a data item may themselves be considered as replicas of their underlying logical write operation. That is, if some transaction writes x at some security level, then all copies of x at higher security levels must be written as well. Thus each *logical* write operation generates many *physical* operations.

We should note that if one wishes to consider issues of transaction integrity, subjects are usually taken to be sequences of these operations. This slightly more complex definition could be used here as well. However since it is more cumbersome and adds virtually nothing to the model for our purposes, we will use this simpler definition.

What is the security level of a physical data item? There are two reasonable possibilities. First, the data item security level could be that of the backend database where it appears. Second, the data item security class could be could be that of the underlying logical data item of which it is a replica. Fortunately, we need not choose. Both are used but for different purposes. The physical security level of the data item is the confidentiality label for the data item. The logical security level label provides information which is necessary for correct use of that data, i.e., for integrity purposes. Notice that the physical security level of a data item always dominates its logical security level. The SINTRA system is assumed to maintain the logical security level of a data items as data within the backend database which contains it.

Similar considerations hold for operations. An operation has a security level determined by the level of the user who submitted it. In addition, write operations spawn other write operations at higher security levels. Thus operations also have two associated security levels; a logical one determined by the user, and a physical one determined by the backend database in which the operation is actually executed.

In the SINTRA architecture, the simple security and restricted ⋆-properties can be replaced (in the sense of permitting user access to data) with the following functionally equivalent statements.

1) A *read* operation can access a data item if and only if its security level equals the physical security level of the data item.

2) A *write* operation can access a data item if and only if its physical security level equals the physical security level of the data item and its logical security level equals the logical security level of the data item.

An implication of this is that operations can access only data items in the physical database at their own security level. Notice that editing a database requires both read and write accesses.

4. Formalizing the SINTRA Security Policy

The model includes the following elementary items

S = a finite set of subjects

O = a finite set of objects

L = a finite lattice of security levels with ordering $\geq$

$A = \{read, write\}$, a set of access modes

C = an equivalence relation on O

U = an equivalence relation on S

$g_p : S \rightarrow L$, a mapping of subjects to (physical) security levels

$f_p : O \rightarrow L$, a mapping of objects to (physical) security levels

W = a predicate on S

O represents the data items of the SINTRA system, including all replicas, and xCy means that x and y are replicas of the same logical data item. S represents the subjects and sUt means that s and t are the same logical operation on copies, possibly different, of two copies of the same logical data item. W is used to distinguish the write operations of S, and Ws means that s represents a write operation. We denote the equivalence classes of x and s in C and U, respectively, by $C(x)$ and $U(s)$.

These elementary items satisfy the following conditions

1) $\forall x \forall y: xCy \wedge f_p(x)=f_p(y) \ .\Rightarrow.\ x=y$

2) $\forall x \forall a \forall b: f_p(x)=a \wedge b>a \ .\Rightarrow.\ \exists y\ (xCy \wedge f_p(y)=b)$

3) $\forall x \forall y \forall a \forall b: f_p(x)=a \wedge f_p(y)=b \wedge xCy \ .\Rightarrow.\ \exists c\ (b>c \vee a>c) \vee a=b$

4) $\forall s \forall t: sUt \wedge g_p(s)=g_p(t) \ .\Rightarrow.\ s=t$

5) $\forall s \forall a \forall b: Ws \wedge g_p(s)=a \wedge b>a \ .\Rightarrow.\ \exists t\ (Wt \wedge sUt \wedge g_p(t)=b)$

6) $\forall s \forall t \forall a \forall b: g_p(s)=a \wedge g_p(t)=b \wedge sUt \ .\Rightarrow.\ \exists c\ (b>c \vee a>c) \vee a=b$

7) $\forall s: \neg Ws \wedge sUt \ .\Rightarrow.\ s = t$

These conditions guarantee that the model accurately reflects the informal model discussed earlier. Conditions 1 and 4 insure that there is at most one replica of a data item or operation at any given security level. Conditions 2 and 5 require that if a data item or write operation exists at a security level then it is replicated at all higher security levels. Conditions 3 and 6 require replicas with possibly noncomparable security levels to arise from the existance of lower security level replicas. Condition 7 eliminates the need to actually replicate read operations.

We define another mapping $f_l : O \rightarrow L$ of objects to (logical) security levels as follows

$$f_l(x) = \text{greatest lower bound } \{a \mid xCy \wedge f_p(y)=a\}$$

Similarly, we define the mapping $g_l : S \rightarrow L$ of subjects to (logical) security levels as follows

$$g_l(s) = \text{greatest lower bound } \{a \mid sUt \wedge g_p(t)=a\}$$

Notice that $f_p(x) \geq f_l(x)$ for any $x \in O$ and that xCy implies that $f_l(x) = f_l(y)$. Moreover, for each $C(x)$ there is a unique minimal object $x_0 \in C(x)$ for which $f_p(x) = f_l(x)$. The object x_0 "represents" the logical data item associated with the equivalence class of x in C. Virtually identical remarks hold for g_l, g_p, and $U(s)$. There are a number of other interesting properties which relate these entities to the SINTRA architecture which we leave to the reader.

Following [9], a *system* is a state machine with states $V \subseteq \wp(S \times O \times A)$, where $\wp(X)$ is the power set of X. A subject s has access a to object o if and only if $(s,o,a) \in v \in V$. The set of requests to change a system state is denoted by R. The state transition mapping $T: R \times V \rightarrow V$ represents the change of state when request r is issued in state v. A system $\sum(V, T, v_{init})$ over a set of requests R is the finite state machine with states V, transition mapping T, and initial state v_{init}. A state v is *reachable* in a system $\sum(V, T, v_{init})$ if there is a sequence $<(r_0,v_0), \cdots, (r_n,v_n)>$ for which $v_0 = v_{init}$ and for all $0 \leq i < n$, $T(r_i,v_i) = v_{i+1}$.

A state v is *read secure* if and only if $(s,x,read) \in v \Rightarrow f_p(x) = g_p(s)$. A state is *write secure* if and only if $(s,x,write) \in v \Rightarrow f_l(x) = g_l(s)$ and $f_p(x) = g_p(s)$. A system is *secure* if and only if all of its reachable states are both read secure and write secure. This corresponds to the definition of system security given in [9,10].

An alternative, but equivalent, way of approaching system security for those types of systems under consideration here can be obtained by slightly reorganizing these conditions. A state is *confidentiality secure* if and only if for an operation *op*, where *op* is either a read or write operation, $(s,x,op) \in v \Rightarrow f_p(x) = g_p(s)$. A state is *integrity secure* if and only if $(s,x,write) \in v \Rightarrow f_l(x) = g_l(s)$. A system is *RA-secure* (replicated architecture-secure) if and only if all of its reachable states are both confidentiality secure and integrity secure.

We assert that a system is secure if and only if it is RA-secure. The proof follows trivially from the definitions.

5. Security and The SINTRA Architecture

We consider the SINTRA architecture database from the point of view of RA-security. The TFE is trusted to connect the user to only the backend database with the security which matches that of the user. Any transactions, and therefore any read or write operations, submitted by a user and which have the security level of that user can act only on data items whose physical security level matches that of the user. It is transparent, then, that the SINTRA system is confidentiality secure. This means that that SINTRA architecture prohibits low security level users from gaining access to high security level information. Hence, in the sense of confidentiality, the SINTRA system has strong security properties. In fact, because of the physical separation of the backend databases, covert channels can arise only through faults in the TFE processes which, in addition to providing trusted separation of access by users, must also manage database functions related to transaction management, recovery, garbage collection, etc. Dealing with covert channels in these processes involves use of trusted mechanisms, minimization of covert channel bandwidths, auditing, etc., but is beyond the scope of this paper whose focus is on the confidentiality obtained by physical separation.

We also argue that the SINTRA architecture database is reasonably integrity secure, though the argument is less apparent. What we must argue is that write operations can be prevented from writing data items whose logical security level is dominated by the security level of the write operation. Most databases provide mechanisms for enforcing constraints on operations based on values of the data. These mechanisms would be utilized to prevent actions which would violate integrity security. If the logical security level labels, which are stored as data, are subject to corruption, e.g., by Trojan Horse attacks, then the non-label data would be equally corruptible, rendering the database functionally useless. The integrity security of the system is enforced as strongly as the integrity of the other data in the system. Thus providing integrity security properties for a SINTRA architecture database are equivalent to that of providing integrity in any database, trusted or otherwise.

6. Extensions of the Model

There are other considerations in the security of the SINTRA architecture database beyond those related to access control discussed above. The access control model must be integrated with that for the trusted portion of the TFE, which we have assumed to have its own security policy formal model, but that is relatively easy. In addition, there are issues involving the means by which the consistency of the replicas of a logical data item will be maintained. These are related to control of concurrent execution of user-submitted transactions and transaction atomicity. There have been a number of concurrency control algorithms developed for this architecture which address these issues[2,3,4,6,7,8]. All of these are presented in a sufficiently formal framework that integrating them into the model presented here is not difficult, though the presentation would be quite lengthy and dependent on the choice of particular algorithm.

Another plausible extension of the model presented here is to non-database applications. Virtually nothing in the access control model is specific to databases. Any application for which the objects to which access is to be controlled are suitable for replication, e.g., files, messages, can be modeled analogously. Obviously, the issue of replica consistency integrity must be resolved adequately in each case.

7. Conclusions

We have given an example of a trusted application based on a replicated architecture and developed a formal model of access control. We have demonstrated that for such systems confidentiality integrity is achieved solely as a function of the architecture, and that integrity security is as good as the integrity of the backend systems' data. Based on this, we conclude that the replicated architecture approach is a simple, straightforward, and promising approach to the development of trusted applications.

References

1. D.E. Bell and L.J. LaPadula, "Secure Computer Systems:Unified Exposition and Multics Interpretation," MTR-2997, The Mitre Corp., March 1976.

2. Oliver Costich, "Transaction Processing Using an Untrusted Scheduler in a Multilevel Database with Replicated Architecture", in *Database Security V:Status and Prospects*, ed. Sushil Jajodia and Carl Landwehr, North-Holland, 1992.

3. Oliver Costich and John McDermott, "A Multilevel Transaction Problem for Multilevel Secure Database Systems and its Solution for the Replicated Architecture", *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA May 1992.

4. Oliver Costich and Myong Kang, "Maintaining Multilevel Transaction Atomicity in MLS Database Systems with Replicated Architecture", in *Database Security VII:Status and Prospects*, ed. Thomas Keefe and Carl Landwehr, North-Holland, 1994.

5. Judith N. Froscher and Catherine Meadows, "Achieving a Trusted Database Management System using Parallelism," in *Database Security II:Status and Prospects*, ed. Carl Landwehr, pp.151-160, North-Holland, 1989.

6. Sushil Jajodia and Boris Kogan, "Transaction Processing in Multilevel-Secure Databases using Replicated Architecture" in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 360-368, Oakland, CA May 1990.

7. Myong Kang, Oliver Costich, and Judith Froscher, "A Practical Transaction Model and Untrusted Transaction Manager for a Multilevel Secure Database System", *Database Security VI:Status and Prospects*, eds. Sushil Jajodia and Carl Landwehr, North-Holland, 1993.

8. John McDermott, Sushil Jajodia, and Ravi Sandhu, "A Single Level Scheduler for the Replicated Architecture for Multilevel-Secure Databases" in *Proceedings of the Seventh Annual Computer Security Applications Conference*, San Antonio, Tx December 1991.

9. John McLean, "Specifyng and Modeling Computer Security", *IEEE Computer* 23(1) pp. 9-16 (January 1990).

10. John McLean, "Security Models", in *Encyclopedia of Software Engineering*, John Wiley & Sons, Inc., New York, (forthcoming).