

**Implementation of Mandatory Access
Control in Role-based Security System with Oracle Snapshot Skill**

CSE 367 Independent Study Final Project Report

Prof. Steve Demurjian

December, 13 2001

Hui Wang

Lisa Yan

{hwang, hyan}@engr.uconn.edu

Computer Science & Engineering

The University of Connecticut

Storrs, CT 06269-3155

Contents

Abstract

1. Introduction

2. Technical Overview of Mandatory Access Control System

2.1 Mandatory Access Control System

2.2 Bell-Lapadula Model

Figure 1: Accessibility of different levels of users and resources

2.3 Multilevel secure (MLS) database management systems

Figure 2: non-classified multi-sensitivity level database model.

3. Government classified information in physically separate machines

Figure 3: users and resources in physically separate Machines (no networking)

Figure 4: MAC system in military application

4. Distribute database of MAC system with Oracle Snapshot skill

4.1 Architecture of MAC distribute database project

Figure5: architecture of our project.

4.2 Database requirements:

4.3 Oracle Read-Only Snapshot skill

Figure 6. Read-only snapshot architecture

4.4 Read-only Snapshot advantages

5. Implementation Steps

5.1. Define database and method level

Figure 7: Assign tables and methods to four classified levels.

Figure8: Ideal architecture of MAC project.

5.2. Make Snapshots

Figure 9: MAC distributed database implementation architecture.

5.3 Change Java code

6. Conclusions and Future Work

7. Attachments

Abstract

Due to sensitive resources, traditional government classified information systems store data on physically separated machines. Each machine cannot communicate with others of different levels, which prevents updating resources efficiently. In this paper, we present a new constraint-base security model implement distributed database to enhance sensitive data security of Mandatory Access Control security system (MAC), and increase communication among each distributed site in current government classified information system and achieve data replication.

1. Introduction

The U.S. government has been involved in developing security technology for computer and communications systems for some time. Although advances have been great, it is generally perceived that the current state of security technology has, to some extent, failed to address the needs of all[1][2]. This is especially true of organizations outside the Department of Defense[3].

The current set of security criteria, criteria interpretations, and guidelines has grown out of research and development efforts on the part of the DoD over a period of twenty plus years. Today, the best known U.S. computer security standard is the Trusted Computer System Evaluation Criteria (TCSEC [4]). It contains security features and assurances, exclusively derived, engineered and rationalized based on DoD security policy, created to meet one major security objective - preventing the unauthorized observation of classified information. The result is a collection of security products that do not fully address security issues as they pertain to unclassified sensitive processing environments. Although existing security mechanisms have been partially successful in promoting security solutions outside of the DoD[2], in many instances these controls are less than perfect, and are used in lieu of a more appropriate set of controls.

The TCSEC specifies two types of access controls: Discretionary Access Controls (DAC) and Mandatory Access Controls (MAC). Since the TCSEC's appearance in December of 1983, DAC requirements have been perceived as being technically correct for commercial and civilian government security needs, as well as for single-level military systems. MAC is used for multi-level secure military systems, but its use in other applications is rare. The combination of MAC and Role-Based Access Control (RBAC), which can be more appropriate and central to the secure processing needs within

industry and civilian government, will be talked in detail in Jin Ma's report. The premise of our project is a new MAC database architecture, which is more secure and fits the TCSEC security criteria.

In section 2 we will review the theoretic background of Bell-Lapadula model (MAC). We also introduce a multilevel secure (MLS) database management system, which is an application that applies MAC idea, and points out problems and consideration. Section 3 proposes government-classified information in physically separated machines, addresses MAC rules (Read Down/Write Up), and points out problems and considerations. Section 4 introduces a new architecture which implements distributed database to enhance sensitive data security of MAC, and add communication among each distributed machine and achieve replication for current government classified information system. Section 5 introduces the implementation steps and Section 6 is the conclusion, recommendations and future work in this project. Section 7 is about attachment files.

2. Technical Overview of Mandatory Access Control System

2.1 Mandatory Access Control System

There are two basic types of access control mechanisms used to protect information from unauthorized access: discretionary access controls (DAC) and mandatory access controls (MAC).

Access controls that are not based on the policy are characterized as discretionary controls by the U.S. government and as need-to-know controls by other organizations. The latter term connotes least privilege — those who may read an item of data are precisely those whose tasks entail the need. Because DAC places the decision of who can access information at the discretion of the creator of the information, DAC is not applicable to the majority of health care information.

Mandatory Access Control, or MAC, relies on labels that correspond to the sensitivity levels of information for clients and objects.

MAC policy compares the sensitivity label at which the user is working to the sensitivity label of the object being accessed and refuses access unless certain MAC checks are passed. MAC is *mandatory* because the labeling of information happens automatically, and ordinary users cannot change labels unless an administrator authorizes them.

Sensitivity labels are assigned to files, devices, windows, hosts, networks, and to other system objects that users access. Administrators indicate the level of trust or job responsibility of anyone accessing the system by assigning a clearance that sets the upper bound of a set of sensitivity labels at which the user can work. Administrators also assign a minimum sensitivity label that sets the lower bound. Alternately, administrators can configure users to work at a single label. With mandatory controls,

only administrators and not owners of resources may make decisions that bear on or derive from policy. Only an administrator may change the category of a resource, and no one may grant a right of access that is explicitly forbidden in the access control policy. MAC requires all those who create, access, and maintain information to follow rules set by administrators.

The restrictions placed on file manipulation (reading, writing, creating, etc.) are those that are generally accepted when implementing a MAC policy:

1. To read a file, the label of the process must dominate the label of the file.
2. To write to a file, the label of the process must be dominated by the label of the file. A process can only create a file to the level of the label.

For example, a user who is running a process at Secret should not be allowed to read a file with a label of Top Secret. Conversely, a user who is running a process with a label of Secret should not be allowed to write to a file with a label of Confidential.

The access decisions to read (query) objects and write (alter) objects are determined by a general concept of equivalence and dominance between the label of a process (subject) and the label of an object (file, directory, etc.). Defining dominance is left to the conforming implementation, but generally a label "dominates" another label if it is "equal or higher" in some defined structure. For example, in military terms, a label of Top Secret dominates a label of Secret. To read an object, the label of the subject must dominate the label of the object.

Current Mandatory Access Control is based on Bell-Lapadula Model.

2.2 Bell-Lapadula Model

Bell-LaPadula Model was proposed by Bell and LaPadula for enforcing access control in government and military applications. The subjects and objects are partitioned into different security levels. A subject can only access objects at certain levels determined by his security level. Follows Write Up/Read Down rules. For instance, the following are two typical access specifications: Unclassified personnel cannot read data at confidential levels and Top-Secret data cannot be written into the files at unclassified levels.[12]

Bell-LaPadula model supports mandatory access control by determining the access rights from the security levels associated with subjects and objects. Mandatory access control is "a means of restricting access to objects based on the sensitivity (as represented by a label) of the information

contained in the objects and the formal authorization (e.g., clearance) of subjects to access information of such sensitivity”[11]

Each object has different access rights that are grouped by role name, and the use of resources is restricted to individuals authorized to assume the associated role. The set of classification levels can be the set (top-secret, secret, confidential, unclassified). For instance, top-secret, {Nuclear, NATO} dominates secret, {NATO}. Top-secret can also access NATO, but Secret can’t access Nuclear. See Figure1 [5].

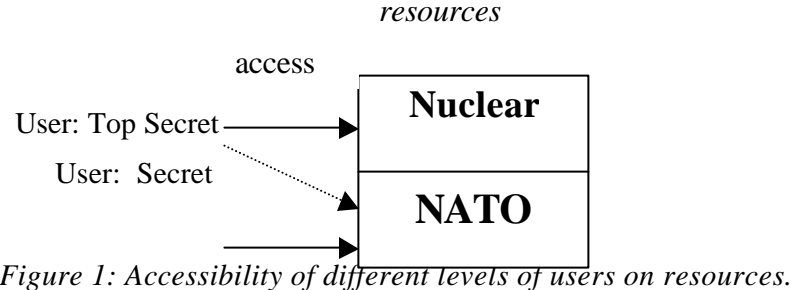


Figure 1: Accessibility of different levels of users on resources.

2.3. Multilevel secure (MLS) database management systems

Mandatory Access Controls (MAC) systems are appropriate for many multilevel secure applications (MLS). Multilevel secure is the implementation of MAC idea application. And we introduced it as the traditional MAC application.

Multilevel database systems is attempting to develop database systems that protect classified information from unauthorized users based on the classification of the data and the clearances of the users. The data stored in a local database system are classified into several levels. Access rights are grouped by level, and the use of resources is restricted to individuals authorized to assume the associated level.

At the conceptual level, a database that contains data labeled over a set of sensitivity levels has relations that may contain data labeled over this same set of sensitivity levels. These multilevel relations are decomposed into single-level or system-high fragments. The multilevel secure DBMS stores the fragments within physically separate single-level objects. Then, the MLS DBMS can enforce mandatory access control on requests to access these separate single-level or system-high objects.

The following Figure2 is a multi-sensitivity level database model:

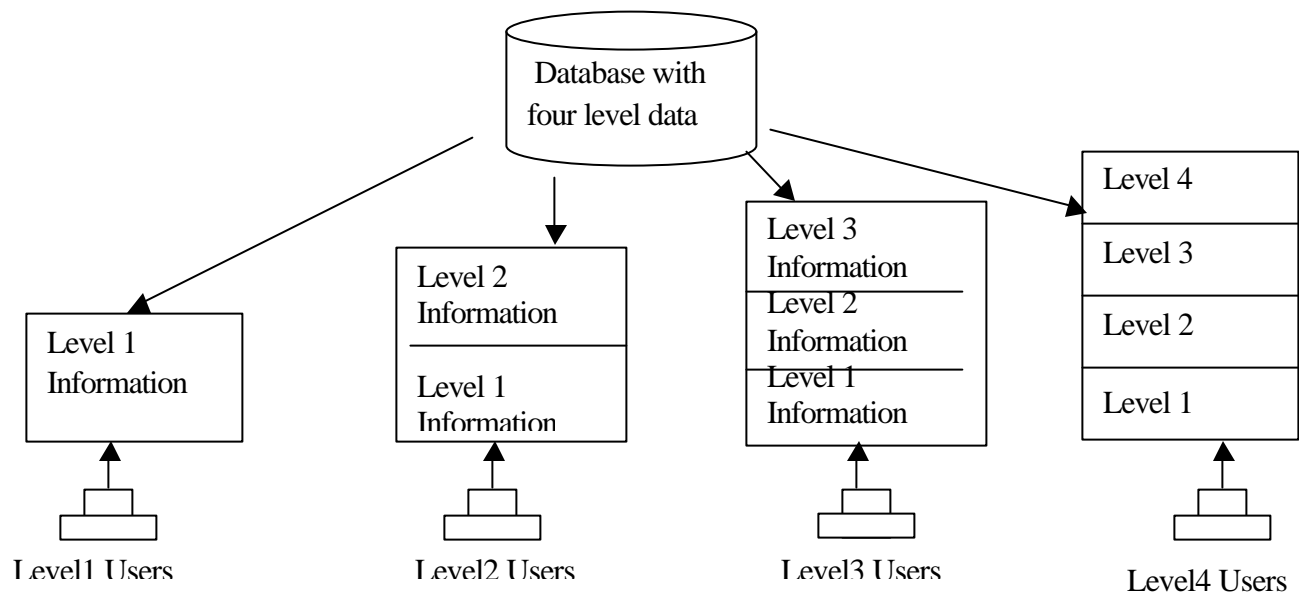


Figure 2: non-classified multi-sensitivity level database model.

Although much work has been done related to security concern in Multilevel secure (MLS) DBMS, when the system processing very sensitive data, it is still contains certain limitations. Due to different authorization level, one can access the same database site, this centralized database lacks flexibility and security. One of the problems is the inference problem when dealing with sensitive data. An inference channel in a database is a means by which one can infer data classified at a high level from data classified at a low level. The inference problem is the problem of detecting and removing inference channels. An inference of sensitive data from nonsensitive data can only be represented within a database if the nonsensitive data itself is stored in the database.[13]

Although MAC is appropriate for many multilevel secure applications, due to lack of security of MLS database management systems and high security requirements of government classified information, government classified information system turns to use Bell LaPadula Model to store different level data into physically separate machines. Next, we are going to talk about government security for roles and resources in a physically separate environment.

3. Government classified information in physically separate machines

Government security information systems provide a uniform system for classifying, declassifying, and safeguarding national security information. It recognizes that the interests of the United States and its citizens require that certain information concerning the national defense and foreign relations be protected against unauthorized disclosure.

Following Bell LaPadula Model, Government security information is classified into the following three levels: ‘Top Secret’, ‘Secret’, and ‘Confidential’. ‘Top Secret’ includes the information that the unauthorized disclosure of the information reasonably can be expected to cause exceptionally grave damage to the national security. ‘Secret’ applies to information that the unauthorized disclosure of the information reasonably can be expected to cause serious damage to the national security. ‘Confidential’ points to the information that unauthorized disclosure of which reasonably can be expected to cause damage to the national security. Except, as other wise provided by statute, no other terms shall be used to identify classified information [14].

Current applications of Government classified information are implemented in physically separate machines. Figure 3 shows the current Government classified information system architecture in non-networking situation. Machine 1,2 and 3 are physically separated. There are no communications among each other.

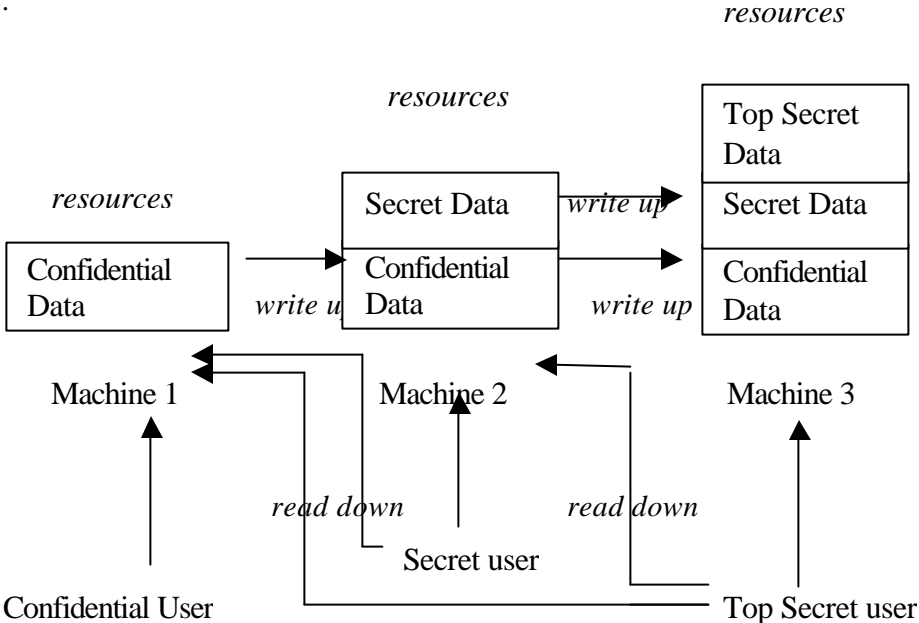


Figure 3: users and resources in physically separate Machines (no networking)

Due to no networking between machines, each machine has to work separately. If Confidential User needs to update Confidential data on Confidential Machine. This user will have to log on the other two machines to repeat the update twice because Secret and Top Secret User have no right to update Confidential data on their own machines. It is a time consuming and resource wasting process to update the same data three times on each machine. However high security requirements of government classified data prevents us from utilizing MLS database management systems on a centralized database environment.

The following Figure 4 is an example model of current security control of MAC systems in military applications:

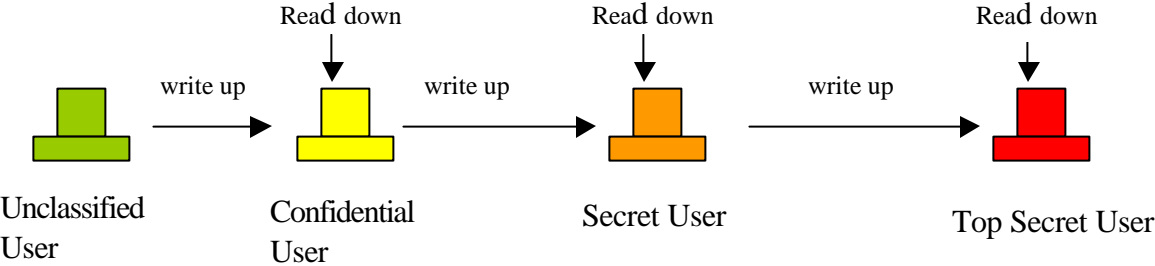


Figure 4: MAC systems in military application

Detail explanation of example model:

Label: The MAC system labels each machine, and information stored on that machine is labeled as the same level.

Read down and write up: a high level machine can display information from low level machine, for example, a disk copied from a low level machine, or an e-mail sending from a low level machine.

Limitation: There are limited communications between each machine. A high level user on a HL(high level) machine can not access a low level machine at all, and can not send any information such as an e-mail to any machine or user in low level.

It is very clear that the current MAC system is not considering dominance. Generally a label "dominates" another label if it is "equal or higher" in some defined structure. In military terms, a label of Top Secret dominates a label of Secret. By considering domination, a high level user should have privilege to access a low level machine. The limitation of the current system also limits the communication between each level.

4. Distribute database of MAC system with Oracle Snapshot skill

4.1 Architecture of MAC distribute database project

In order to match government classified information need, in this section, we propose a new architecture by using basic Snapshot skill to accomplish replication and concurrency security control in distributed database system. This new architecture distributes a database to enhance sensitive data security of MAC system, add communications with each distributed machine, and achieves replication for current government classified information system.

The new architecture was not built from scratch. It is a modified current government classified information system architecture and still follows Bell-LaPadula Model. Figure 5 shows the basic architecture for our project:

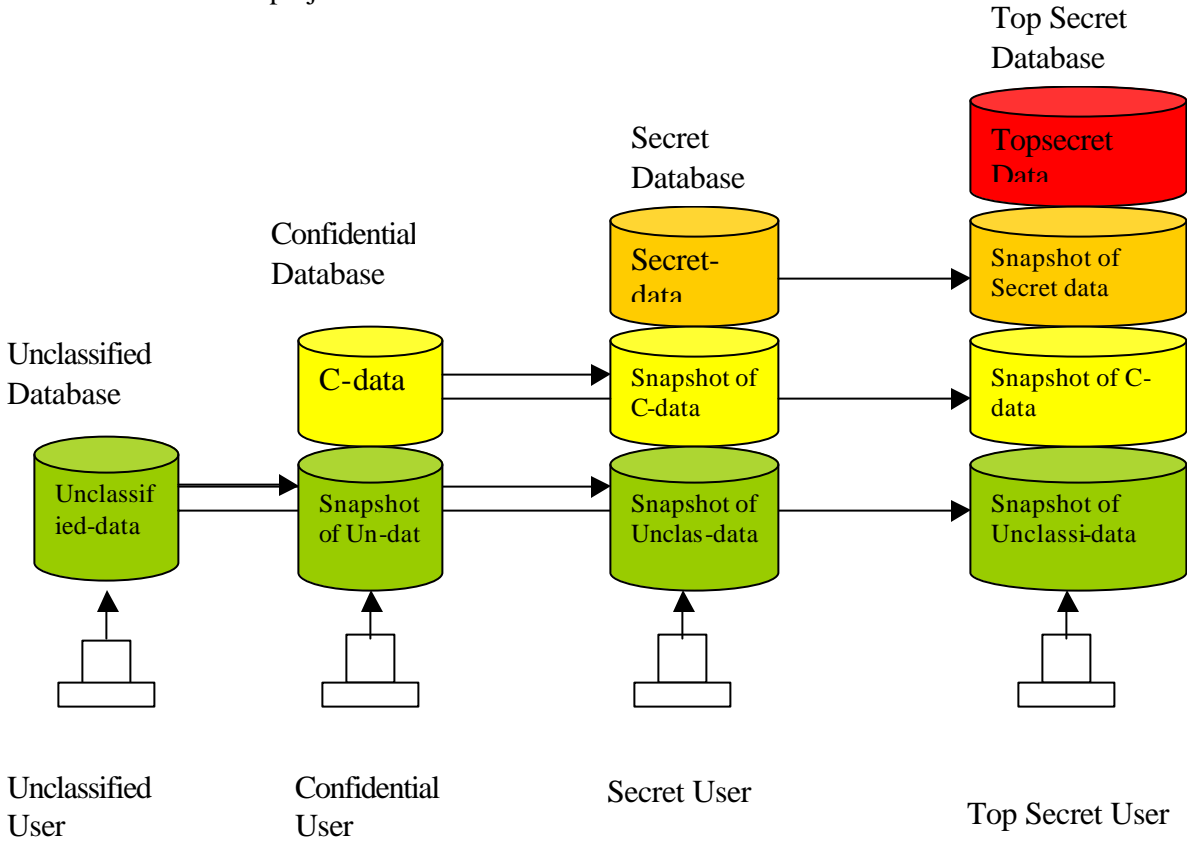


Figure 5: architecture of our project.

Explanation of this architecture:

1. Mandatory Access Control, or MAC, relies on labels that correspond to the sensitivity levels of information. Users who are using MAC system are classified into four levels: Unclassified, Confidential, Secret and Top Secret.
2. MAC policy compares the sensitivity label at which the user is working to the sensitivity label of the object being accessed and refuses access unless certain MAC checks are passed. MAC is *mandatory* because the labeling of information happens automatically, and ordinary users cannot change labels. The system will judge the user's level and lead the user to access the corresponding database.
3. *Database*: there are four database sites and they are labeled into four levels: Unclassified, Confidential, Secret and Top Secret, and each site is on a machines with the same level.

4. *Write up*: for users in each level, they have full privilege to read and write to the data source labeled as the same security level. When they write to the data, a copy of the data is sent to the database site labeled with higher security level.
5. *Read down*: A user with a higher level can read any information of the lower level from the same security level database site, yet they cannot make changes on this information.
6. *Dominance*: in our system, we implement dominance concept from the military. A user from high level dominates low-level information. It does not mean he/she can do any change of low-level information on the database site with the same level. To make any changes, he/she has to access the low level database to make the changes.
7. Sensitivity labels are assigned to database objects that users access. Administrators indicate the level of trust or job responsibility of anyone accessing the system by assigning a clearance that sets the upper bound of a set of sensitivity labels at which the user can work. Administrators also assign a minimum sensitivity label that sets the lower bound. Alternately, administrators can configure users to work at a single label. (See detail at Jin Ma's paper)

4.2 Database requirements:

From this architecture, there are following requirements on the database structure:

1. *Distribute*: TCSEC specifies that data of different levels cannot be on one machine. We have to distribute our databases on machines with different security levels.
2. *Accessibility*: User of different level can only edit data of the same level.
3. *Write up*: When a lower level user works on the same level data, the changes they make will reflect on databases of the higher levels.
4. *Read down*: User on high level can read low level data on high level but can not edit the lower level data on their database site.
5. *Dominance*: High level users use the following ways to follow the dominance requirement: they can access low level database site and make changes on the low level data. (Jin Ma will fulfill this requirement in her part by assigning different level methods to different roles).

We use oracle snapshot skill to implement the database replication to fulfill MAC database requirements perfectly.

4.3 Oracle Read-Only Snapshot skill

Oracle replication skills include: read-only snapshots, updateable snapshots, single master replication, multimaster replication and deployment templates. We choose read-only snapshot skill to finish our project.

Oracle uses *snapshots*, also referred to as *materialized views*, to replicate data to non-master sites in a replicated environment and to cache "expensive" queries in a data warehouse environment.

A snapshot is a replica of a target master table from a single point in time. Snapshots are updated from one or more master tables through individual batch updates, known as a *refreshes*, from a single master site. To keep a snapshot's data relatively current with the data of its master table, the snapshot must be periodically refreshed. A *snapshot refresh* is an efficient batch operation that makes a snapshot reflect a more current state of its master table.

Snapshots can also contain a WHERE clause so that snapshot sites can contain customized data sets. Such snapshots can be helpful for regional offices or sales forces that do not require the complete corporate data set.

When a snapshot is refreshed, Oracle must examine all of the changes to the master table from a log file to see if any apply to the snapshot. Therefore, if any changes were made to the master table since the last refresh, a snapshot refresh will take some time, even if the refresh does not apply any changes to the snapshot. If, however, no changes at all were made to the master table since the last refresh of a snapshot, the snapshot refresh should be very quick.

A read-only snapshot is essentially a local table whose data is refreshed at specified intervals by performing a query against one or more remote tables. The inventory application can create the same functionality as the database link described in the previous section by following these steps:

1. Create database link between two Oracle site
2. Create Snapshot log on the master table.
3. Create Snapshot on the replication site target at the master table on the master site.

Snapshots use the Oracle built-in package DBMS_JOB to schedule snapshot refresh tasks.

The following is the Oracle Read-Only Snapshot architecture.

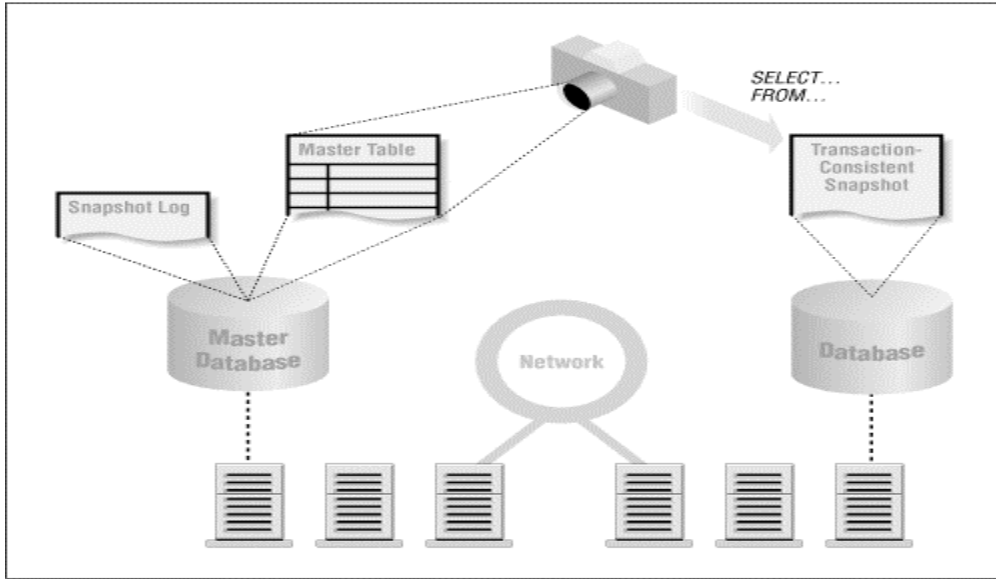


Figure 6. Read-only snapshot architecture

4.4 Read-only Snapshot advantages

By using Read-only Snapshot skill, there are following advantages:

- **Keep Sensitive Data Secure:** Users can only view data that satisfies the defining query for the snapshot. On MAC system, users on each level can only have access privileges to the data of the corresponded level, and users in high level can only read data of lower levels on their site. If the high level users want to do some change on low level data, they have to access the low level site to make changes. By using read-only snapshot skill, the requirements can be fulfilled.
- **Keep data consistency:** Snapshots follow the predefined schedules do refreshments. This advantage guarantees that snapshots keep the consistency with the master table.
- **Reduce Network Traffic:** Only changes that satisfy the snapshot's WHERE clause of the defining query are propagated to the snapshot site, thereby reducing the amount of data transferred and reducing network traffic. Snapshots are updated through an efficient batch process from a single master site. They have lower network requirements and dependencies than multimaster replication because of the point in time nature of snapshot replication. Whereas multimaster replication requires constant communication over the network, snapshot replication requires only periodic refreshes. In addition to not requiring a dedicated network connection, replicating data with snapshots increases data availability by providing local access to the target data. These benefits, combined with mass deployment and data subsetting (both of which also

reduce network loads), greatly enhance the performance and reliability of your replicated database.

5. Implementation Steps

We have already explained how Oracle Read-Only Snapshot skill fulfills the database requirement of Mandatory Access Control. In this section, we will introduce how we apply Oracle Read-Only Snapshot skill on MAC project.

Our MAC project is based on Role-based security system project (See detail at Dan Wang and Fei Gao' paper [16]). There are two resource databases: university database and patient database. We only apply us ideas on patient database. The following are implementation steps:

5.1. Define database and method level

There are four tables in patient database: *patient*, *payment*, *medicalHistory* and *prescription*. There are ten methods target at these four databases and are supposed to be assigned to three kinds of roles:

For doctor:

```
writeDiagnosis( );  
writePrescription( );
```

For doctor and nurse:

```
getDiagnosis ( );  
getPrescription ( );  
getMedicalHistory();
```

For accountant:

```
getPaymentMode( );  
setPaymentMode( );  
addPatient( );  
removePatient( );  
getPatientList( );
```

We assign four tables and corresponded methods to four classified levels as Figure 7:

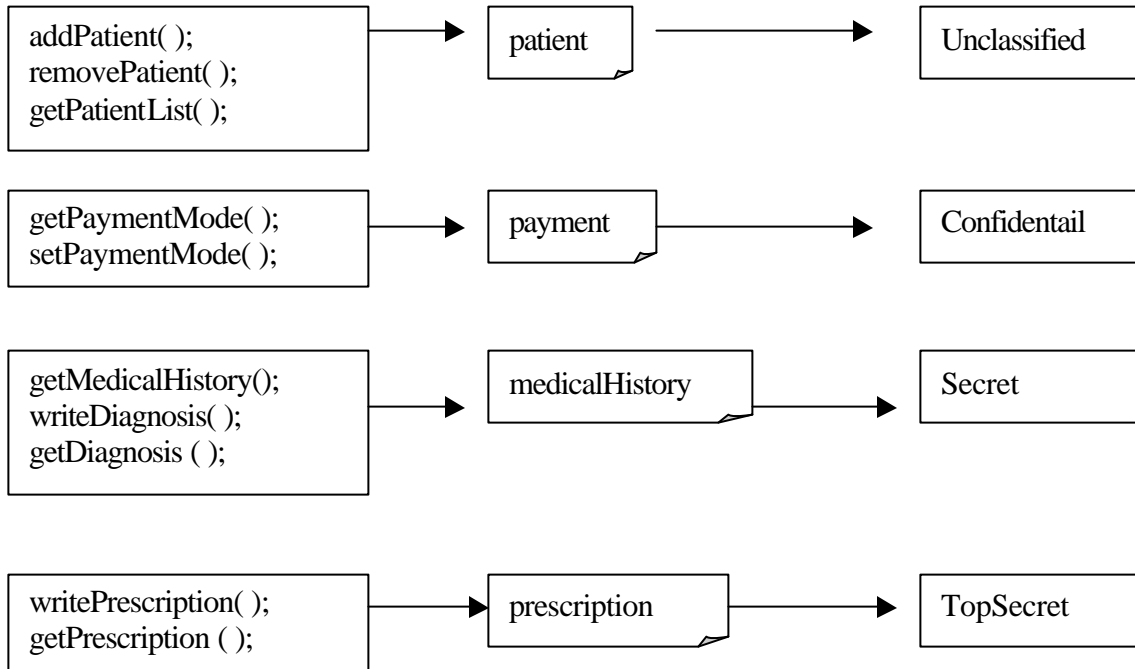


Figure 7: Assign tables and methods to four classified levels.

The ideal MAC distributed database structure follows the architecture in figure 8. There are four Oracle database site, each site is classified into different level. For each level database site, there are stored correspondent classified data, and low level read only data. Low level read only data is implemented by Oracle Read-Only Snapshot skill.

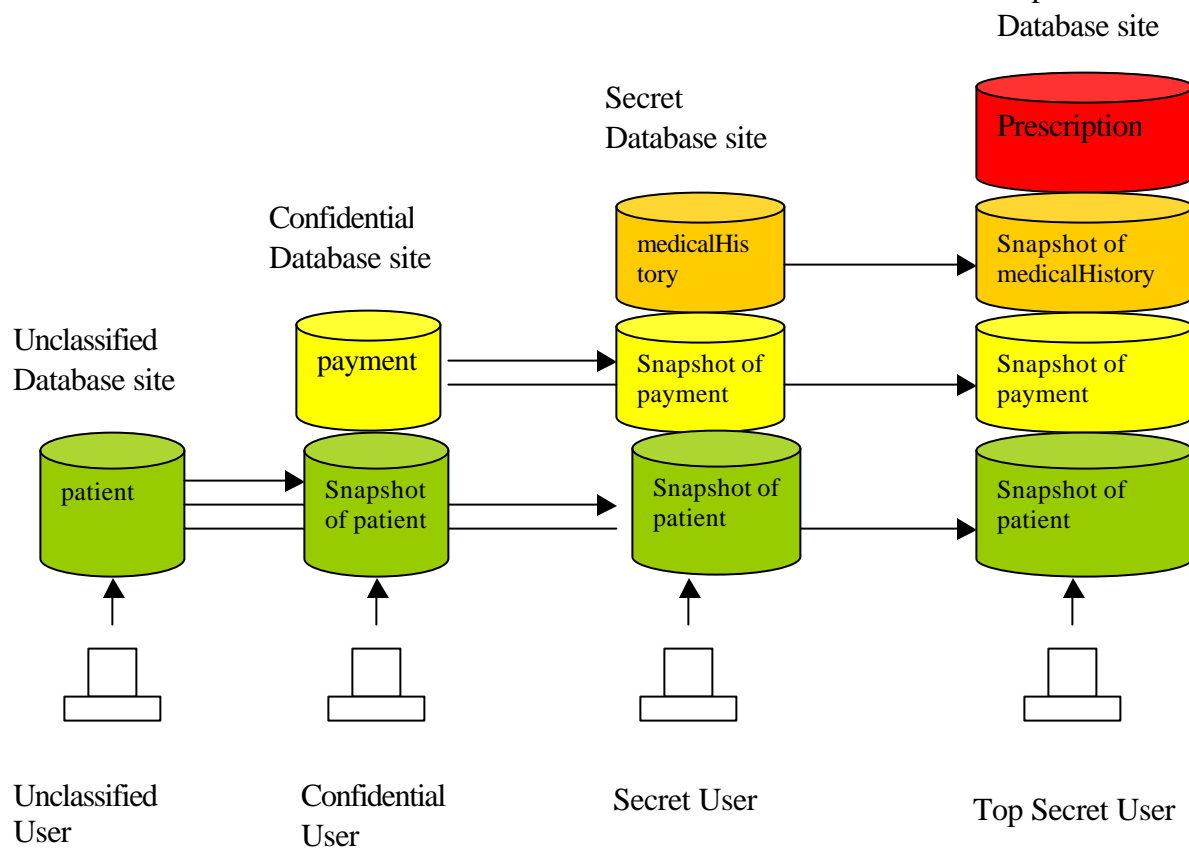


Figure8: Ideal architecture of MAC project.

5.2. Make Snapshots

It is ideal to have four Oracle site to implements our ideas, but we have only three Oracle sites. The three sites are: CSE367(installed in Linux machine), KOKOU and DRAGON(both installed in NT machine). To implements our ideas on these three machines, we have to put a Secret database and a Top Secret database on one machine, and create two databases for them. Because we can not make snapshots between two users of one database site, we use another approach: Grant select privilege to Top Secret user on table medicalHistory. After the grant, the Top Secret user can read from medicalHistory but can not make any changes. It still fulfills our MAC security requirement.

Our Oracle sites are on a same network, and our resource tables are limited, so we set our snapshot refresh time period as one minute. Each snapshot will do refresh every minute.

Figure 9 is the final implementation architecture. For snapshot implementation detail, please read our attached *.sql files.

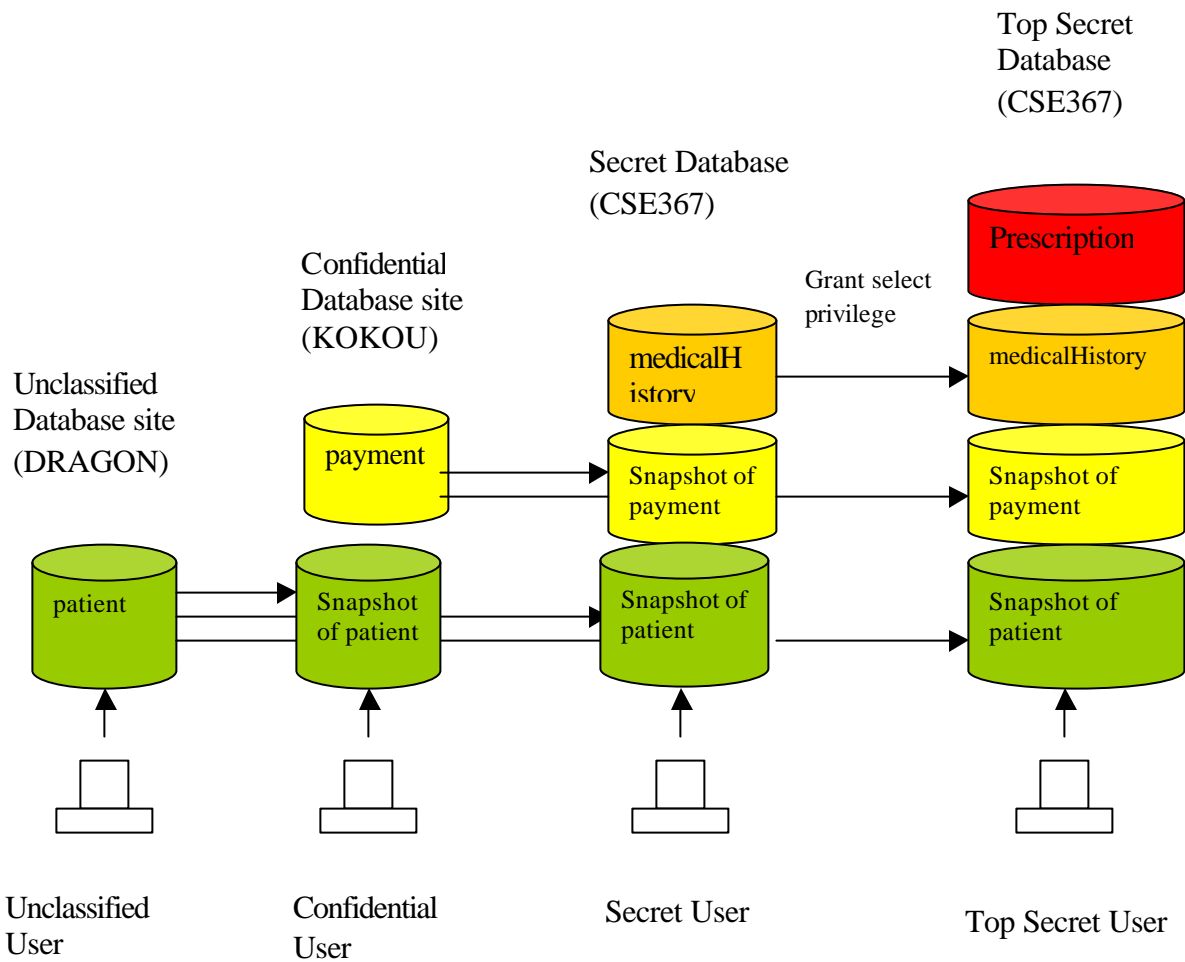


Figure 9: MAC distributed database implementation architecture.

5.3 Change Java code

After we set up our database design, we made corresponding changes in the original Java code. We wrote comments in the code we changed. The following are the changes we made based on the original version.

1. src/corbass/common/Const.java
Put new database connect information
2. src/corbass/pdbServer/PDBCorbaliInterfaceImpl.java
Rewrote each method, assigned them to different Oracle database sites.
3. src/corbass/pdbServer/PDBResourceID.java

Assigned method level. Through this java file, resource is connected with security part(Jin Ma's work). Security administrator assigns the correct method to different level of role. The basic idea for this work is: for a user clarified as higher level, not only the methods labeled with the same level are assigned to the user, but also all methods labeled bellows the current label. MAC uses this idea to assure the security dominance requirement.

6. Conclusions and Future Work

We designed a new architecture for MAC applications that fits the TCSEC security requirements and MAC requirements very well. We also implemented this idea successfully with the Oracle Read Only Snapshot skill. We compared previous MAC applications, which require manual update on database to our design which uses scheduled regular refreshments, and found it more efficient and secure.

Although Oracle Read Only Snapshot fits all of our security requirements very well, it is not a real time replication. But real time is not reasonable in a situation requiring multiple database updates and replication. We didn't test our implementation with Oracle sites on different networks with large amounts of resources. MAC constraints can be used on facilities like printer and scanner, or even on application software. We will leave them as our future work.

Because we based our project on a previous Role-Based Security system, the security level we assigned to each method may not sound reasonable or practical, but the concepts are good.

7. Attachments:

1) snapshot_Test_problems.sql

This file contains a pretest snapshot procedure. We record all problems we encounter and solutions.

2) cse367_snapshot.sql, kokou_snapshot.sql, dragon_snapshot.sql

These three files contain all procedures we did on three sites.

References

1. D.D. Clark and D.R. Wilson. A Comparison of Commercial and Military Computer Security Policies. In *IEEE Symposium on Computer Security and Privacy*, April 1987.
2. Computers at Risk. National Research Council, National Academy Press, 1991.
3. Minimum Security Functionality Requirements for Multi-User Operating Systems (draft). Computer Systems Laboratory, NIST, January 27 1992.
4. Trusted Computer Security Evaluation Criteria, DOD 5200.28-STD. Department of Defense, 1985.
5. T.C. Ting, S.A. Demurjian, and M.Y. Hu. *Requirements Capabilities and Functionalities of User-Role Based Security for an Object-Oriented Design Model*. In C.E. Landwehr and S. Jajodia, editors, *Database Security V: Status & Prospects*, pages 275-96. North-Holland, 1992
6. D. E. Denning and P.J. Denning. *Certification of Programs for Secure Information Flow*. ACM, 20(7):504-13, July 1977
7. R.S. Sandhu. *Lattice-based access control models*. Computer, 26:9-19, Nov. 1993
8. Krishnamurthy and McGuffin. *On the Design & Administration of Security Database Transactions*. ACM SIGSAC Review, pages 63-70, Spring/Summer 1992
9. Charles Dye. *Oracle Distributed System*. O'reilly , April 1999.
10. Jensen, Kiel and Verjinski. *SDDM a Prototype of A Distributed Architecture for Database Security*. In Proc. Of Int'l Conf. On Data Eng., pages 356-64, Feb 1989
11. Morrie Gasser. *Building a Secure Computer System*. Van Nostrand Reinhold Company New York, 1988 ISBN 0-442-23022

12. Matunda Nyanchama. *Modeling Mandatory Access Control in Role-Based Security System*. IFIP Workshop on Database Security. 1995
13. Bell, D. E. and LaPadula, L. J. *Secure Computer Systems: Unified Exposition and Multics Interpretation*, MTR-2997 Rev. 1, MITRE Corp., Bedford, Mass., March 1976.
14. Alexander Brodsky. *Secure Databases: Constraints, Inference Channels, and Monitoring Disclosures*. IEEE, Vol. 12, No. 6 Nov/Dec 2000
15. DoD Directive 5210.10. *Unauthorized Disclosure of Classified Information to the Public*. Feb. 1992, ASD(C3I)
16. Fei Gao and Dan Wang. *Implementation of Constraint-based Security Model In CORBA and Jini Environment*. Computer Science and Engineering department , the University of Connecticut.