An Introduction to Multilevel Secure Relational Database Management Systems

Walid Rjaibi

IBM Toronto Software Laboratory
Markham, Ontario, Canada
wrjaibi@ca.ibm.com

Abstract

Multilevel Security (MLS) is a capability that allows information with different classifications to be available in an information system, with users having different security clearances and authorizations, while preventing users from accessing information for which they are not cleared or authorized. It is a security policy that has grown out of research and development efforts funded mostly by the U.S. Department of Defense (DoD) to address some of the drawbacks of the single level mode of operation that was used at the DoD. The goal was to build and deploy an MLS-compliant environment (e.g., Networks, Operating Systems, Database Systems) that would provide a much needed efficiency in processing and distributing classified information by providing security through computer security, communications security, and trusted system techniques instead of using physical controls, administrative procedures, and personnel security. As Relational Database Management Systems (RDBMS) are at the heart of the DoD’s information system, significant research and development efforts have been put into building multilevel secure RDBMS, which have led to the emergence of a number of multilevel secure RDBMS solutions, including commercial ones. Over the past few years and with the increase of security concerns, MLS compliance has become a major requirement from a number U.S. Federal Government agencies that appear to have grown beyond the traditional agencies that require such type and level of security. This paper introduces MLS, and outlines the challenges and complexities of building a multilevel secure RDBMS. The paper also gives concrete examples of both research and commercial multilevel secure RDBMS and describes how they met the above challenges and complexities.

1 Introduction

Multilevel Security (MLS) is a capability that allows information with different classifications to be available in an information system, with users having different security clearances and authorizations, while preventing users from accessing information for which they are not cleared or authorized[2]. It is a security policy that has been developed primarily for the U.S. military and intelligence communities, but has also been adopted by some civilian organizations that store, process and distribute classified information (e.g., major aircraft manufacturers) as well as by a number of defense departments around the world.
Given the extremely high value of the information that could be stored in a military or intelligence database, and the potential damage that could result from the unauthorized disclosure, alteration or loss of such information, preventing users from accessing information for which they are not cleared or authorized requires much more than just implementing an access control policy. In particular, security guards must be put in place to prevent users from gaining access to information for which they are not cleared or authorized through indirect means.

Covert channels[5] are examples of such indirect means. A covert channel can easily be established with conventional database concurrency control algorithms such as two-phase locking (2PL) and timestamp ordering (TO)[6]. In both 2PL and TO algorithms, whenever there is contention for the same data item by transactions executing at different security levels, a lower level transaction may be either delayed or suspended to ensure correct execution. In such a scenario, two colluding transactions executing at high and low security levels can establish an information flow channel from a high security level to a low security level by accessing selected data items according to some agreed-upon code[4].

Inference[7] is another indirect means by which users can gain knowledge about information for which they are not cleared or authorized. For example, enforcing a primary key constraint[6] across data from different security levels could allow a non sufficiently cleared user to gain knowledge about the existence of a data row at a higher security level from the duplicate key error message that is returned to that user when he or she attempts to insert a data row at a low security level but having the same primary key as the data row at the higher security level.

Building a multilevel secure RDBMS has thus posed significant challenges to the database research community. For instance, secure database transaction protocols had to be developed, and a solution to reconcile the conflicting requirements between data integrity and confidentiality had to be found. MLS has also posed significant challenges to database vendors as building a multilevel secure RDBMS often requires rebuilding major portions of an existing commercial RDBMS.

There has been an abundance of research within the last two decades or so in the area of multilevel secure RDBMS. Such research has addressed specific aspects of building a multilevel secure RDBMS such as secure transaction protocols, system architectures, or polyinstantiation[8], and there is a rich set of publications about those specific aspects[8, 4, 9, 10]. However, the multilevel secure RDBMS research literature surprisingly lacks the kind of publication that would allow someone to get a good understanding about what it takes to build a multilevel secure RDBMS as a whole, as well as to serve as a quick guide for those who might be thinking about building such RDBMS.

Moreover, the term multilevel security is heavily overloaded across the Information Technology (IT) industry and often means different things to people from different backgrounds as there are not only multilevel secure RDBMS, but also multilevel secure operating systems, multilevel secure networks, multilevel secure web servers, etc. In addition to being heavily overloaded, MLS is often incorrectly used interchangeably with emerging marketing terms such as Label-Based Access Control (LBAC), Row-Level Security, and others. All of this makes it extremely difficult for those who have not been directly involved in designing or building a multilevel secure RDBMS to get a good understanding about what it really takes to build a multilevel secure RDBMS.

In this paper, the author wishes to share his expertise in database security and privacy to try to clarify the mystery of multilevel security, as well as to outline the challenges and complexities of building a multilevel secure RDBMS.

1.1 Synopsis

The rest of this paper is organized as follows. Section 2 introduces MLS and describes the MLS certification and evaluation process. Section 3 presents and compares Multilevel Secure RDBMS architectures. Section 4 describes the issue of polyinstantiation. Section 5 presents multilevel secure transaction pro-
cessing. Section 6 gives concrete examples of both research and commercial multilevel secure RDBMS. Lastly, section 7 summarizes the concepts introduced in this paper.

2 What is Multilevel Security?

A good understanding of MLS would not be complete without understanding its origins, and what problems it was meant to solve. The U.S. military and intelligence communities have historically segregated data based upon its security classification. Classified data must reside and be processed on dedicated systems that do not provide access to users outside of the immediate community of interest and are often separated by an air gap and connected only by a sneaker net[2]. The main drawbacks of such operational scheme can be summarized as follows:

- Redundant databases: To store data with different security levels (e.g., Top Secret data and Unclassified data), a separate database must be created and maintained for each security level.
- Redundant workstations: A user who is required to access data with different security levels (e.g., Top Secret data and Unclassified data) would be required to use a different workstation to access each type of data.
- High cost of IT infrastructure: It is not possible to share the computer and communication system infrastructures, such as cabling, network components, printers, and workstations without risking to compromise security.
- Inefficiency: Staff members need to access several systems to perform their duties.

The U.S. DoD has therefore funded significant research and development projects across various organizations to come up with a solution that would allow classified information to be stored, processed and distributed in a secure way, but without the drawbacks listed above. MLS was that solution[2]. MLS allows information with different classifications to be available in an information system, with users having different security clearances and authorizations, while preventing users from accessing information for which they are not cleared or authorized[2]. For example, an MLS system might process both Secret and Top Secret collateral data and have some users whose maximum clearance is Secret and others whose maximum clearance is Top Secret. Another MLS system might have all its users cleared at the Top Secret level, but have the ability to release information classified as Secret to a network consisting of only Secret users and systems. In each of these instances, the system must implement mechanisms to provide assurance that the system’s security policy is strictly enforced. MLS has resulted in a shift from providing security through physical controls, administrative procedures, and personnel security to providing security using computer and communication security.

2.1 The Bell-LaPadula Multilevel Security Model

The Basic model of MLS was first introduced by Bell and LaPadula[11]. The model is stated in terms of objects and subjects. An object is a passive entity such as a data file, a record, or a field within a record. A subject is an active process that can request access to objects. Every object is assigned a classification, and every subject a clearance. Classifications and clearances are collectively referred to as labels. A label is a piece of information that consists of two components: A hierarchical component and a set of unordered compartments. The hierarchical component specifies the sensitivity of the data. For example, a military organization might define levels Top Secret, Secret, Confidential and Unclassified. The compartments component is nonhierarchical. Compartments are used to identify areas that describe the sensitivity or category of the labeled data. For example, a military organization might define compartments NATO, Nuclear and Army. Labels are partially ordered in a lattice as follows: Given two labels $L_1$ and $L_2$, $L_1 \geq L_2$ if and only if the hierarchical component of $L_1$ is greater than or equal to that of $L_2$, and
the compartment component of L1 includes the compartment component of L2. L1 is said to dominate L2. MLS imposes the following two restrictions on all data accesses:

- The Simple Security Property or “No Read Up”: A subject is allowed a read access to an object if and only if the subject’s label dominates the object’s label.
- The *-Property (pronounced the star property) or “No Write Down”: A subject is allowed a write access to an object if and only if the object’s label dominates the subject’s label.

### 2.2 Evaluation and Certification

Multilevel secure systems must complete an extensive evaluation and certification process before they can be used in military applications. The evaluation and certification of a multilevel secure system is usually conducted by an independent testing laboratory and is based upon a clearly defined set of criterion. One set of criteria is called common criteria, which has recently been adopted as an ISO standard[3]. Another set of evaluation criteria used by the U.S. DoD is the Trusted Computer System Evaluation Criteria (TCSEC)[5]. Most multilevel secure RDBMS have been developed before common criteria was adopted. TCSEC has been the norm for evaluating such RDBMS.

TCSEC is divided into four divisions: D, C, B, and A ordered in a hierarchical manner with the highest division (A) reserved for systems providing the most comprehensive security. Each division represents a major increase in the overall confidence, or trust, that one can place in the system. Successive levels of trust build upon and incorporate the criteria of the previous lower level of trust.

Within Divisions C and B there are a number of subdivisions known as classes. The classes are also ordered in a hierarchical manner with systems representative of Divisions C and B characterized by the set of computer security mechanisms that they possess. For Division C, Discretionary Access Control (DAC)[6] is provided, whereby users can grant or deny access by other users and groups of users to the system resources that the users control. For Division B, Mandatory Access Control (MAC)[1] is provided. MAC employs the simple security property and the *-property of the Bell-LaPadula MLS model to protect data of different security levels. Division A also provides the MAC features.

Systems representative of the higher classes in Division B and Division A derive their security attributes more from their design and implementation structure than merely security features or functionality. Increased assurance that the required features are operative, correct, and tamperproof under all circumstances is gained through progressively more rigorous design, implementation, and analysis during the development process. Division A requires formal (e.g., mathematical) design and verification techniques to provide increased assurances over Division B.

Multilevel secure systems are associated with TCSEC divisions B and A[2].

### 3 Multilevel Secure RDBMS Architectures

Multilevel secure RDBMS architectures can be divided into two general types, depending on whether mandatory access control is enforced by the RDBMS itself or delegated to a trusted operating system. These two general types are the Woods Hole Architecture and the Trusted Subjects Architecture[9, 10].

#### 3.1 Woods Hole Architectures

The Woods Hole architectures are the outcome of a three-week study on trusted data management sponsored by the U.S. Air Force at Woods Hole, Massachusetts, USA in 1982[9, 10]. The subject of this study was the following: Can we build a multilevel secure RDBMS using existing untrusted off-the-shelf RDBMS, with minimal change?

The Woods Hole architectures assume that an untrusted off-the-shelf RDBMS is used to access data and that trusted code is developed around that RDBMS to provide an overall secure RDBMS. They can be divided into two main categories: The kernelized architectures and the distributed architectures[9, 10].
3.1.1 Kernelized Architectures

The kernelized architecture[9, 10] uses a trusted operating system and multiple copies of an off-the-shelf RDBMS, where each copy is associated with some trusted front-end. Each pair (trusted front-end, RDBMS) is associated with a particular security level. The trusted operating system enforces its full access control policy on all accesses by the RDBMS to the RDBMS objects. It ensures that data at different security levels is stored separately, and that each copy of the RDBMS gets access to data that is authorized for its associated security level. The latter is possible because the multilevel database is decomposed into multiple single-level databases, where each represents a fragment of the conceptual multilevel database. Each fragment is stored in a single-level operating system object (e.g., a file) which is labeled by the operating system at the corresponding security level, and thus can only be accessed according to the MAC policy of the operating system.

Figure 1 illustrates a kernelized architecture where one RDBMS is associated with the security level “High” and another RDBMS is associated with the security level “Low”. The RDBMS associated with the security level “High” has access to both the fragment of the database at the high security level and the fragment of the database at the low security level. But the RDBMS associated with the security level “Low” has access only to the fragment of the database at the low security level.

A benefit of this architecture is that data at different security levels is isolated in the database, which allows for higher level assurance. Another benefit is that, assuming an already evaluated operating system, this architecture should minimize the amount of time and effort to evaluate the RDBMS. However, this architecture results in an additional overhead as the trusted operating system needs to separate data at different security levels when it is added to the database and might also need to combine data from different security levels when data is retrieved by an RDBMS copy that is associated with a high security level.

3.1.2 Distributed Architectures

The distributed (or replicated) architecture[9, 10] is a variation of the kernelized architecture. It uses multiple copies of the trusted front-end and RDBMS, each associated with its own database storage. In this architecture scheme, an RDBMS at security level \( l \) contains a replica of every data item that a subject at level \( l \) can access. Thus, when data is retrieved, the RDBMS retrieves it only from its own database. Another benefit of this architecture is that data is physically separated into separate hardware databases. However, this scheme results in an additional overhead when data is updated as the various replicas need to be kept in sync.

3.2 Trusted Subjects Architectures

The trusted subject architecture[9] is a scheme that contains a trusted RDBMS and a trusted operating system. According to this architecture, the mandatory access control policy is enforced by the RDBMS itself. Database objects (e.g., a table) are stored in operating system objects (e.g., a file) labeled at the highest security level. A database table can con-
tain rows with different security levels. Such rows are distinguished based on their security level which is explicitly stored with each row. This architecture is called “trusted subject” because the RDBMS is privileged to violate the operating system’s MAC policy when accessing database objects. For example, when a user with a low security level queries a database table, the operating system’s object where that table is stored ends up being accessed, which is a violation of the operating system’s MAC policy. But the RDBMS is trusted to return to the users only those rows for which he or she is authorized according to the MAC policy. Figure 2 illustrates a multilevel secure trusted subject RDBMS Architecture.

Figure 2: Multilevel secure trusted subject RDBMS architecture.

A benefit of this architecture is that the RDBMS has access to all levels of data at the same time, which minimizes retrieval and update processing. However, this architecture results in a special purpose RDBMS that requires a large amount of trusted code to be developed and verified along with the normal RDBMS features. It also lacks the potential to be evaluated to high TCSEC evaluation classes because meeting higher levels of assurance requires the ability to provide separation of mandatory objects by some form of hardware isolation. It is also difficult to prove that the trusted software used to isolate mandatory objects (e.g., data rows with different security levels) is working correctly without allowing for the flow of data with high security level to users with low security level.

### 4 Polyinstantiation

Multilevel secure RDBMS utilize mandatory access control to prevent the unauthorized disclosure of high-level data to low-level users. It is also necessary to guard against the threat to confidentiality that can arise from enforcing database integrity constraints[6] across data from multiple security levels. To illustrate this threat to confidentiality, consider the following database table where the attribute “starship” is the primary key, and the attribute “label” represents the data row security level.

<table>
<thead>
<tr>
<th>Starship</th>
<th>Destination</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise</td>
<td>Mars</td>
<td>High</td>
</tr>
</tbody>
</table>

Suppose that a user with a low security level wishes to insert the tuple (Enterprise, Talos, Low). From a purely database perspective, this insert must be rejected because it violates the primary key constraint. However, rejecting this insert could be sufficient to compromise security as the user with low security level could infer that the starship Enterprise is on a mission with a higher security level.

Polyinstantiation[8] is a solution to this problem. It expands the notion of primary key to include the security level so that more than one tuple may possess the same apparent primary key if they are at different security levels. To continue with our example, a new row with the same apparent primary key (i.e., Enterprise) is added to the table.

<table>
<thead>
<tr>
<th>Starship</th>
<th>Destination</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise</td>
<td>Mars</td>
<td>High</td>
</tr>
<tr>
<td>Enterprise</td>
<td>Talos</td>
<td>Low</td>
</tr>
</tbody>
</table>

From a security perspective, the newly added row is simply a cover story for the real mission of the starship enterprise.
In addition to protecting against inference, polyinstantiation is also useful to prevent denial of service to legitimate users as well as to protect against storage covert channels \[5\]. Covert channels use system variables and attributes to signal information. To illustrate this type of threat to confidentiality, consider the following database table where the attribute “starship” is the primary key, and the attribute “label” represents the data row security level.

<table>
<thead>
<tr>
<th>Starship</th>
<th>Destination</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise</td>
<td>Talos</td>
<td>Low</td>
</tr>
</tbody>
</table>

Now, suppose that a user with a high security level wishes to update the destination to be “Mars”. If the RDBMS rejects this update, then the user may have been denied legitimate privileges. If the update is allowed by changing the row’s security level to “High” then a user with a low security level will notice that the data row has disappeared and will infer that its security level has been increased. If the update is allowed without changing the row’s security level, then a storage covert channel will be created. That is, the data row itself could be used as a storage object for passing high level information to users with low security level. Polyinstantiation allows the RDBMS to insert a new data row with the same apparent primary key (i.e., Enterprise) but with a high security level as a result of such update.

<table>
<thead>
<tr>
<th>Starship</th>
<th>Destination</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise</td>
<td>Mars</td>
<td>High</td>
</tr>
</tbody>
</table>

From a security perspective, the old data row is simply a cover story for the real mission of the starship enterprise.

5 Multilevel Secure Transaction Processing

Multilevel secure RDBMS utilize mandatory access control to prevent the unauthorized disclosure of high-level data to low-level users. It is also necessary to guard against the threat to confidentiality that can arise from employing conventional transaction protocols such as two-phase locking (2PL)\[4\]. The 2PL transaction protocol delays the execution of conflicting operations by setting locks on data items for read and write operations\[6\]. A transaction must acquire a shared-lock (S-lock) on a data item before reading it and an exclusive lock (X-lock) before writing it. The 2PL transaction protocol is inherently vulnerable to timing covert channels which could be established to leak confidential information. A timing covert channel \[5\] varies the amount of time to complete a task to signal information. To illustrate this threat to confidentiality, consider the following example.

Let $T_i$ denote a high security level transaction, which is reading a low security level data item $A$. Let $T_j$ denote a low security level transaction, which is trying to write to data item $A$. If the 2PL transaction protocol is employed, then $T_j$ must wait to acquire an X-lock on data item $A$ (i.e., wait until $T_i$ releases its S-lock on data item $A$). Suppose that $T_j$ can measure the time quantum $q$ it has to wait to acquire the lock on data item $A$: A quantum of waiting time greater than a certain amount represents ‘1’, and a quantum of waiting time less than that a certain amount represents ‘0’. Transaction $T_i$ can exploit this knowledge to send one bit of high security level information to $T_j$, and by repeating this protocol, any information can be sent, creating a timing covert channel.

2PL, and in general conventional transaction protocols in RDBMS, are not secure against timing covert channels.

6 Commercial and Research Multilevel Secure RDBMS

The research and development efforts in the area of multilevel secure RDBMS have resulted in a number of commercial and research systems. The most noticeable of these systems are the following: Trusted Oracle\[12\], Informix OnLine/Secure\[13\], Sybase Secure SQL Server\[14\], DB2 for z/OS\[15\], Trusted
Rubix[16], SEAVIEW[8], and Unisys Secure Distributed DBMS[17].

Trusted Oracle can be configured to run in one of two modes: DBMS MAC and OS MAC. The former is an architecture where mandatory access control is enforced by the RDBMS itself, and thus is a trusted subject architecture. The latter is a kernelized architecture (i.e., mandatory access control is delegated to the operating system). Informix OnLine/Secure, Sybase Secure SQL Server, DB2 for z/OS, and Trusted Rubix are examples of a trusted subject architecture. The SEA VIEW research system is an example of a kernelized architecture whereas the Unisys Secure Distributed research RDBMS is an example of a distributed architecture.

Informix OnLine/Secure, Sybase Secure SQL Server, Trusted Oracle, and Trusted Rubix all support polyinstantiation. The key for a tuple in Informix OnLine/Secure automatically includes the tuple security label. Thus, polyinstantiation is always possible and cannot be suppressed by the RDBMS.

The tuple security label in the Sybase Secure SQL Server is part of all keys. Thus, polyinstantiation is always possible and cannot be suppressed by the RDBMS.

Trusted Oracle can be configured to run in one of two modes. When run in DBMS MAC mode, a single Trusted Oracle database can store information at multiple security levels. In this mode, Trusted Oracle can turn polyinstantiation on and off at the table level by requiring key integrity which does not include the tuple security label. When on, the primary key includes the tuple label, which allows polyinstantiation to occur. When off, the key does not include the tuple security label, thus preventing polyinstantiation.

When run in OS MAC mode, Trusted Oracle is capable of storing data at only a single security label, and the RDBMS is constrained by the underlying operating system MAC policy. Without any MAC privilege, the RDBMS cannot suppress polyinstantiation because a low RDBMS will not be aware of any tuple with the same primary key at a higher security level, and a high RDBMS cannot be trusted to modify data at a low security level. As such, polyinstantiation cannot be prevented when Trusted Oracle is running in OS MAC mode.

Informix OnLine/Secure and Trusted Oracle provide secure transaction processing protocols. Informix OnLine/Secure uses an approach by which a transaction at a low security level can acquire a write lock on a low data item even if a transaction at a high security level holds a read lock on that data item. Thus, a transaction at a low security level is never delayed by a transaction at a high security level. The transaction at the high security level simply receives a warning that a lock on a low data item has been “broken”. Trusted Oracle uses a combination of locking and multiversioning techniques.

7 Conclusion

This paper has given an overview of multilevel security, the MLS evaluation and certification process, and multilevel secure RDBMS. Building a multilevel secure RDBMS can be a challenging task. Depending on the architecture followed, this might require rebuilding major portions of an existing commercial RDBMS. It also requires significant effort to evaluate and certify, particularly if a high level of assurance is sought. We are not aware of any commercial RDBMS that has been evaluated higher than B1 according to the Trusted Computer Security Evaluation Criteria.

Mandatory access control, polyinstantiation, and secure transaction processing are the key aspects of a multilevel secure RDBMS. However, these are not sufficient to ensure that security cannot be compromised. Depending on how stringent the requirements of the organization that wishes to deploy a multilevel secure RDBMS, the RDBMS might have to implement additional security guards. For example, SQL compilers have traditionally been guided by performance reasons in selecting the order in which the predicates contained in a query are evaluated (i.e., more selective predicates are often evaluated first to narrow down the set of rows to be passed on to a subsequent join because join operations are costly). If the method chosen to enforce MAC when accessing a table is based on query modification to incorporate the MLS two security properties in
the form of regular predicates, then special care must be taken in selecting the order in which the predicates on that table are evaluated to avoid unauthorized leakage of data rows. To illustrate how leakage could occur, suppose that a query has a predicate on a table that involves a User-Defined Function (UDF). Further suppose that this UDF takes the whole data row as an input parameter and that the UDF source code makes a copy of the data row outside the database (or sends it as an e-mail to some destination). Now, assume that some data row R cannot be returned to the user who issued the query because this would violate the MLS security properties. If the predicate involving the UDF is evaluated prior to evaluating the predicates that implement the MLS security properties then data row R will be consumed by the UDF and consequently leaked to an unauthorized user.

Database triggers[6] are another example where additional security guards could be necessary. A trigger could cause labeled data row to flow from a table on which mandatory access control is enforced to another table on which mandatory access control is not enforced. Without proper flow control measures, triggers could cause unauthorized leakage of information to occur.

Acknowledgements

The author wishes to thank Calisto Zuzarte and Kelly Lyons from the IBM Toronto Laboratory for their suggestion to write a CASCON paper about multilevel secure RDBMS.

Trademarks

IBM and Informix are registered trademarks of International Business Machines Corporation in the United States, other countries, or both. Other company, product and service names may be trademarks or service marks of others.

Disclaimer

The views expressed in this paper are those of the authors and not necessarily of IBM Canada Ltd. or IBM Corporation.

About the Author

Walid Rjaibi joined IBM in 1996. He initially worked at the IBM Toronto Lab within the DB2 UDB query optimization team for five years. In this role, Walid was the architect and author of several innovative solutions including the extensions made to the DB2 statistics model to support parallel database environments and the query performance simulator. Walid then joined IBM Research in Zurich, Switzerland (ZRL) where he worked as a Research Staff Member in Network Security and Cryptography for two years. At ZRL, he was an active member of the IBM Privacy Technology Institute (PTI) where he developed innovative solutions for enabling RDBMS to automatically enforce privacy policies. Walid returned to the IBM Toronto Lab in March 2003 where he joined the newly formed DB2 UDB security development team. He has authored several research and technical papers on database security and privacy, and holds a patents portfolio of eleven filed or granted patents. Walid holds a Computer Engineer degree from the University of Tunis (Tunisia), and a Masters degree in Computer Science from Laval University (Canada).

References


