Distributed Database Security

İlker Köse
Data and Network Security - Spring 2002
GYTE, Computer Engineering

Introduction

Databases have been common in government departments and commercial enterprises for many years. Today, databases in any organization are increasingly opened up to a multiplicity of suppliers, customers, partners and employees - an idea that would have been unheard of a few years ago. Numerous applications and their associated data are now accessed by a variety of users requiring different levels of access via manifold devices and channels - often simultaneously. For example:

- On-line banks allow customers to perform a variety of banking operations - via the Internet and over the telephone – whilst maintaining the privacy of account data.
- e-Commerce merchants and their Service Providers must store customer, order and payment data on their merchant server -and keep it secure.
- HR departments allow employees to update their personal information – whilst protecting certain management information from unauthorized access.
- The medical profession must protect the confidentiality of patient data – whilst allowing essential access for treatment.
- On-line brokerages need to be able to provide large numbers of simultaneous users with up-to-date and accurate financial information.

This complex landscape leads to many new demands upon system security. The global provide mechanisms to segregate environments; perform integrity checking and maintenance; enable strong authentication and non-repudiation; and provide for confidentiality. In turn, this necessitates comprehensive business and technical risk assessment to identify the threats, vulnerabilities and impacts, and from this define a security policy. This leads to security definitions throughout the infrastructure - operating system, database management system, middleware and network.

Financial, personal and medical information systems and some areas of government have strict requirements for security and privacy. Inappropriate disclosure of sensitive information to the wrong parties can have severe social, legal and regulatory consequences. Failure to address the basics can result in substantial direct and consequential financial losses - witness the fraud losses through the compromise of several million credit card numbers in merchants' databases, plus associated damage to brand-image and loss of consumer confidence.

Database Security

Database management systems normally run on top of an operating system and provide the security associated with a database. Typical operating system security features include memory and file protection, resource access control and user authentication. Memory protection prevents

the memory of one program interfering with that of another and limits access and use of the objects employing techniques such as memory segmentation. The operating system also protects access to other objects (such as instructions, input and output devices, files and passwords) by checking access with reference to access control lists. Security mechanisms in common operating systems vary tremendously and, for those that are lacking, there exists special-purpose security software that can be integrated with the existing environment. However, this can be an expensive, time-consuming task and integration difficulties may also adversely impact application behaviors. Most database management systems consist of a number of modules - including database querying and database and file management - along with authorization, concurrent access and database description tables. These management systems also use a variety of languages: a data definition language supports the logical definition of the database; developers use a data manipulation language; and a query language is used by non-specialist end-users.

Distributed Databases

The developments in computer networking technology and database systems technology resulted in the development of distributed databases in the mid 1970s. It was felt that many applications would be distributed in the future and therefore the databases had to be distributed also. Although many definitions of a distributed database system have been given, there is no standard definition. A distributed database system includes a **distributed database management system (DDBMS), a distributed database and a network for interconnection.** The DDBMS manages the distributed database. A distributed database is data that is distributed across multiple databases.

Distributed database system functions include **distributed query management, distributed transaction processing, distributed metadata management and enforcing security and integrity across the multiple nodes.**

Background for Secure Distributed Database Systems

Concepts in Distributed Databases

There are various architectural alternatives for a distributed database system. In one architecture, the control is centralized while the data is distributed. In another architecture, the data as well as control are distributed. Then there is the non-multidatabase approach to designing a distributed database system. In this approach, there are no local database management systems (DBMSs). The DDBMS manages all of the distributed data. Multidatabase architectures are architectures where each local database is managed by a local DBMS and the various DBMSs are connected through a DDBMS. These multidatabase architectures have been studied extensively in the literature. They can be grouped according to whether they are based on tightly coupled or loosely coupled approaches. An orthogonal method is to group the multidatabase systems depending on whether they are based on homogeneous or heterogeneous DBMSs.

The components of a DDBMS include a distributed query processor **(DQP)** that handles distributed queries, a distributed transaction manager **(DTM)** for processing distributed

transactions, a distributed metadata manager (DTM) for managing distributed metadata, a distributed integrity manager **(DIM)** for enforcing integrity constraints across the databases and a distributed security manager **(DSM)** for enforcing security constraints across the databases.

There are two aspects to query processing in a DDBMS: **query transformation and query optimization.** The query transformation process transforms a global query into equivalent fragment queries. This process is performed according to transformation rules. The query optimization process optimizes the query with respect to the cost of executing the query. Transaction management in a DDBMS involves the handling of distributed transactions. By a distributed transaction, we mean a transaction which executes at multiple sites. The portion of the transaction, which executes at a particular site is a sub transaction associated with that site. A coordinator controls the execution of the sub transactions. **Concurrency control techniques** ensure the consistency of the distributed database when transactions execute concurrently. An additional problem of transaction management in a distributed environment is ensuring the consistency of the data in the presence of site and network failures. Various commit protocols have been developed for reliable transaction processing.
Metadata in a distributed environment includes **information about the relations, fragmentation of the relations, allocation of the relations and the local schema information.** It is usually maintained at two levels. One is the global metadata and the other is the local metadata. Local metadata is the metadata associated with the local DBMS. Integrity issues in a DDBMS include enforcing integrity constraints across multiple DBMSs. Security controls include discretionary security such as enforcing access control rules and multilevel security where the data are assigned different sensitivity levels. We discuss both aspects of security in the sections on multilevel security and discretionary security.

Concepts in Secure Databases

Much of the early work on secure databases was on discretionary security. The major issues in security are **authentication, identification and enforcing appropriate access controls.** For example, what are the mechanisms for identifying and authenticating the user? Will simple password mechanisms suffice? With respect to access control rules, languages such as SQL have incorporated **GRANT and REVOKE** statements to grant and revoke access to users. For many applications, simple GRANT and REVOKE statements are not sufficient. There may be more complex authorizations based on database content. Negative authorizations may also be needed. Access to data based on the roles of the user is also being investigated. Numerous papers have been published on discretionary security in databases.

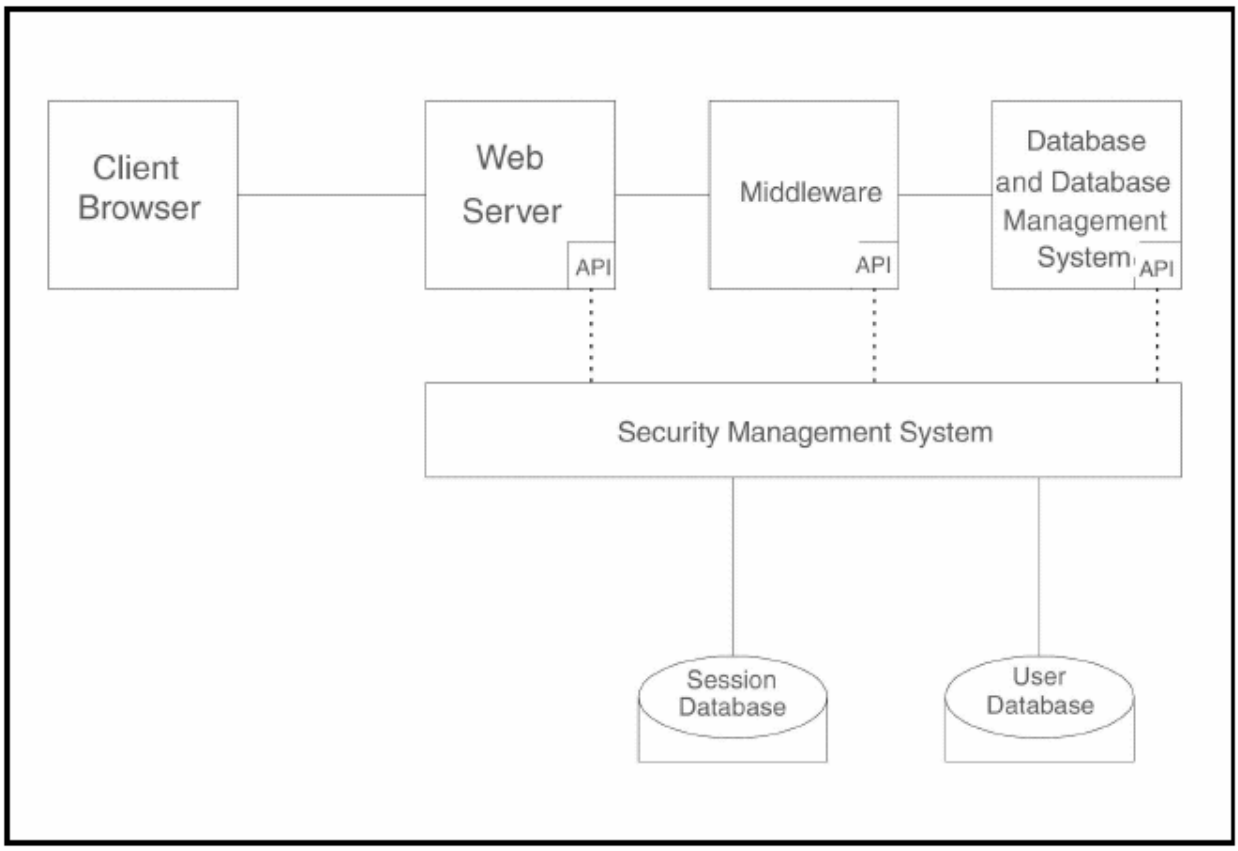

Figure 1: A Simple Model

Database management systems have many of the same security requirements as operating systems, but there are significant differences since the former are particularly susceptible to the threat of improper disclosure, modification of information and also denial of service. Some of the most important security requirements for database management systems are:

- Multi-Level Access Control.
- Confidentiality.
- Reliability.
- Integrity.
- Recovery.

Around 1982, multilevel secure database management system **(MLS/DBMS)** started to be investigated. These systems have also been referred to as Trusted Database Management Systems. In an MLS/DBMS, users cleared at different security levels access and share a database with data at different security levels (also called **sensitivity levels**) without violating security. It is generally assumed that the security levels form a lattice where Unclassified < Confidential < Secret < TopSecret. An MLS/DBMS may also support users and data at different compartments or categories.

The security policy of MLS/DBMSs includes a policy for mandatory access control **(MAC)** and discretionary access control **(DAC)**. Mandatory security controls restrict access to data depending on the sensitivity levels of the data and the clearance level of the user. In general, most MLS/DBMSs enforce a MAC policy where a subject reads an object (such as row or relation) if the subject's security level dominates the security level of the object and a subject updates an object if the subject's security level is that of the object. **Discretionary security** measures are usually in the form of rules, which specify the type of access that users or groups of users may have to different kinds of data. Other components of the security policy include policies for integrity, identification and authentication, auditing and accounting. In general, trusted processes perform security critical functions. These processes must be verified to operate correctly.

While there was some early work in MLS/DBMSs in the 1970s, much of the work in MLS/DBMSs was a consequence of the Air Force Summer Study in 1982. Since then, there have been numerous developments in MLS/DBMSs including the Trusted Database Interoperation (TDI), which interprets the Trusted Computer Systems Evaluation Criteria (TCSEC) for database systems, MLS/DBMS prototypes and designs, and commercial products. There has also been extensive investigation on topics such as the inference problem, secure transaction management and secure database design.

Multilevel Security for Distributed Database Systems

Much of the work in secure distributed database systems has focused on **multilevel security**. Some of the early work in the field began with the Air Force Summer Study. Specifically approaches based on distributed data and centralized control architectures were proposed. Prototypes based on these approaches were also developed during the late 1980s and early 1990s. Notable among these approaches are the efforts by **Unisys Corporation and the Naval Research Laboratory**. Then in the early to mid 1990s, there were efforts at MITRE on secure distributed database systems based on the architecture proposed in [CERI84].

Distributed Data and Centralized Control

The Air Force Summer Study examined a distributed architecture for developing MLS/DBMSs. In this architecture, the data is distributed while control is centralized. Two approaches were proposed at the Summer Study. In the first of these approaches, known as the **Partitioned approach**, a trusted front-end database system is connected to non-trusted back-end database systems. Each back-end database system operates at a single level and manages data at that level. For example, an Unclassified DBMS manages the unclassified data while a Secret DBMS manages Secret data. All communication between the backend database systems is through the front-end database system. A user's query is sent to the DBMS at the user's level or below the user's level. Update requests are sent only to the DBMS at the user's level. For example, a Secret user 's query is sent to the Secret and Unclassified DBMSs, while a Secret user's update request is sent only to the Secret DBMS.

It was found that there was potential for covert channels with this approach as a Secret user's query could have sensitive information that is sent to an Unclassified DBMS. As a result, a second approach was examined **where data is replicated**. In this approach, the unclassified data

is replicated at the Secret and Top-Secret databases, and the Secret data is replicated at the Top-Secret database. This way, a user's query is sent only to the DBMS at the user's level. The update request is sent to the replicated databases. While there is no covert channel for the query operation, since a user's update is propagated upwards, there is a potential for covert channels during the update operation. Nevertheless, this issue has been studied extensively and shown that the bandwidth of the channel is fairly low. Therefore, the preferred approach is the **replicated approach.** This approach has been studied extensively both at Unisys and at the Naval Research Laboratory.

Distributed Data and Distributed Control

In a multilevel secure distributed database management system (MLS/DDBMS), users cleared at different security levels access and share a distributed database consisting of data at different security levels without violating security. This architecture has been derived from the architecture for a DDBMS in [CERI84] and is based on distributed data and distributed control. In this architecture, the MLS/DDBMS **consists of several nodes** that are interconnected by a multilevel secure network. In a **homogeneous environment**, all of the nodes are designed identically. Each node is capable of handling multilevel data. Each node has a MLS/DBMS, which manages the local multilevel database. Each node also has a distributed processing component called the **Secure Distributed Processor (SDP)**.

The modules of the SDP are similar to those of the DDBMS described above. These modules are the Secure Distributed Query Processor **(SDQP)**, the Secure Distributed Transaction Manager (SDTM), the Secure Distributed Metadata Manager (SDMM), the Secure Distributed Security Manager (SDSM) and the Secure Distributed Integrity Manager (SDIM). Multilevel security must be taken into consideration during all of the processing. First of all, an appropriate security policy has to be formulated. This policy will depend on the policies of the local DBMS, the network and the distributed processor. The algorithms for query, update and transaction processing in a DDBMS have to be extended to handle multilevel security. Locking techniques could cause covert channels. Therefore, algorithms for MLS/DBMSs have to be integrated with algorithms for DDBMSs to handle MLS/DDBMSs. These algorithms are implemented by the SDTM. Finally, security and integrity processing techniques for MLS/DBMSs and DDBMSs have to be extended for MLS/DDBMSs. For example, in the case of the SDSM, it has to consider multilevel security within functions such as **identification, authentication and enforcing discretionary security controls.**

Inference Problem

The inference problem has received a great deal of attention for many years. There is still work on this problem especially with emerging technologies such as **data warehousing, data mining and the web.** Inference is the process of users posing queries and deducing unauthorized information from the legitimate responses that they receive. This problem has been discussed a great deal over the past two decades. However, data mining makes this problem worse. Users now have sophisticated tools that they can employ to get data and deduce patterns that could be sensitive. Without these data mining tools, users would have to be fairly sophisticated in their

reasoning to be able to deduce information from posing queries to the databases. In short, data mining tools make the inference problem quite dangerous.

An extensive investigation of the inference problem for distributed database systems began around 1992. This approach was based on processing security constraints in a multilevel secure distributed database system. Essentially, the query processor of the MLS/DDBMS described in the preceding subsection was examined and augmented with constraint processors.

Heterogeneous and Federated Systems

The earliest effort to investigate multilevel security for heterogeneous and federated database systems began around 1991. In particular, various approaches to designing heterogeneous and federated database systems were examined and the security impact was investigated. The issues include **security for integrating the federated schemas as well as handling different ranges of security levels.** For example, one system could handle the range from Secret to Top-Secret, while another system could handle the range from Unclassified to Secret. Issues on integrating heterogeneous database schemas in a secure environment were also investigated. Much of the work here focused on query processing issues. In addition, prototype systems were also developed. The approaches examined both the multidatabase and the nonmultidatabase architectures.

Around 1992, the MITRE Corporation began an effort called MUSET (Multilevel Secure Transactions) database system. This effort focused mainly on transaction management in a heterogeneous database system. This effort also investigated concurrency control, recovery and commit protocols for multilevel transactions. A multilevel transaction is a transaction that can operate at multiple security levels. For example, a sub-transaction at one site could operate at the unclassified level, while another sub-transaction at a different site could operate at the Secret level. The challenge here is to ensure consistency and at the same time minimize covert channels. Other work on secure distributed database systems includes various concurrency control algorithms for transactions.

Discretionary Security

Discretionary security mechanisms enforce rules which specify the types of access that users or groups of users have to the data. Multilevel security controls ensure that users cleared at different security levels access and share a distributed database in which the data is assigned different sensitivity levels without compromising security. In this section, we describe both types of security.

In a distributed environment, users need to be authenticated with respect to the local system as well as the global environment. An issue here is whether the authentication mechanism should be centralized or distributed. If it is centralized, then the authenticator needs to have information about all of the users of the system. If the authenticator is distributed, then the various components of the authenticator need to communicate with each other to authenticate a user.

The access control rules enforced in a distributed environment may be distributed, centralized or replicated. If the rules are centralized, then the central server needs to check all accesses to the database. If the rules are distributed, then appropriate rules need to be located and enforced for a particular access. Often the rules associated with a particular database may also be stored at the same site. If the rules are replicated, then each node can carry out the access control checks for the data that it manages.

Security for Emerging Distributed System Technologies

This section briefly examines various emerging technologies that have evolved in some way from distributed databases and discusses the security impact. These include data warehouses and data mining systems, collaborative computing systems, distributed object systems and the web. First, let us consider data warehousing systems. The major issues here are ensuring that security is maintained in building a data warehouse from the backend database systems and also enforcing appropriate access control techniques when retrieving the data from the warehouse. For example, security policies of the different data sources that form the warehouse have to be integrated to form a policy for the warehouse. This is not a straightforward task, as one has to maintain security rules during the transformations. For example, one cannot give access to an entity in the warehouse, while the same person cannot have access to that entity in the data source. Next, the warehouse security policy has to be enforced. In addition, the warehouse has to be audited. Finally, the inference problem also becomes an issue here. For example, the warehouse may store average salaries. A user can access average salaries and then deduce the individual salaries in the data sources, which may be sensitive and therefore, the inference problem could become an issue for the warehouse. To date, little work has been reported on security for data warehouses as well as the inference problem for the warehouse. This is an area that needs much research.

Data mining causes serious security problems. For example, consider a user who has the ability to apply data mining tools. This user can pose various queries and infer a sensitive hypothesis. That is, the inference problem occurs via data mining. There are various ways to handle this problem. Given a database and a particular data-mining tool, one can apply the tool to see if sensitive information can be deduced from legitimately obtained unclassified information. If so, then there is an inference problem. There are some issues with this approach. One is that we are applying only one tool. In reality, the user may have several tools available to him. Furthermore, it is impossible to cover all of the ways that the inference problem could occur. Another solution to the inference problem is to build an inference controller that can detect the motives of the user and prevent the inference problem from occurring. Such an inference controller lies between the data-mining tool and the data source or database, possibly managed by a DBMS. Data mining systems are being extended to function in a distributed environment. These systems are called distributed data mining systems. Security problems may be exacerbated in distributed data mining systems. This area has received very little attention.

Other emerging technologies that have evolved in some way from distributed databases are **collaborative computing systems, distributed object management systems and the web**. Much of the work on securing distributed databases can be applied to securing collaborative computing systems. With respect to distributed object systems security, there is a lot of work by the Object Management Group's Security Special Interest Group. More recently, there has been

much work on securing the web. The main issue here is ensuring that the databases, the operating systems, the applications, the web servers, the clients and the network are not only secure, but are also securely integrated.

Conclusion

Distributed database systems are a reality. Many organizations are now deploying distributed database systems. Therefore, we have no choice but to ensure that these systems operate in a secure environment. We believe that as more and more technologies emerge, the impact of secure distributed database systems on these technologies will be significant.

References

1. *Paul Lothian and Peter Wenham,* Database Security in Web Environment, 2001
2. *B. Thuraisingham,* Security for Distributed Database Systems, *Computers & Security*, 2000.
3. *Simon Wiseman, DERA,* Database Security: Retrospective and Way Forward, 2001