ISSN 2278-8808
An International Peer Reviewed
SCHOLARLY RESEARCH JOURNAL
FOR INTERDISCIPLINARY STUDIES

SECURITY ISSUES IN OBJECTED ORIENTED DATABASE FOR DISTRIBUTED ENVIRONMENT

Amjad Mahmood Hadi
Iraq Almuthanna – Alsamawa

Received: 09 January 2013
Reviewed & Received: 12 Feb 2013
Accepted: 12 January 2013

Abstract

Object oriented techniques are getting popular in these days and people are moving towards this technology, database management system is one of the largest field in business environment also today's business environment has an increasing need for distributed database System as the desire for reliable, efficient, scalable and accessible information is steadily rising. Distributed database systems provide an improvement on communication and data processing due to its data distribution throughout different network sites. Not only is data access faster, but a single-point of failure is less likely to occur. Security becomes more complex in distributed object oriented database system; this paper discusses the selection of security issues in object oriented database for distributed environment.

Keyword: *Security Issues, Objected Oriented Database, Distributed Environment*

Introduction

An **object database** (also **object-oriented database management system**) is a database management system in which information is represented in the form of objects as used in object-oriented programming. Object databases are different from relational databases and belong together to the broader database management system. Object databases have been considered since the early 1980s and 1990s. Object databases' main usage is in object oriented areas.

Distributed database is a database which logically belongs to the same entity but physically distributed in different sites connected by networks.DDB technology combines both distribution and integration. The distribution aspect is provided by distributing the data across the sites in the network. While the integration aspect is provided by logically integrating the distributed data so that

it appears to the users as a single, homogenous DB. Centralized DB, by contrast, requires both logical and physical integration.

Now a days DDBMS becomes more popular, the need for improvement in distributed database management systems becomes even more important. A distributed system varies from a centralized system in one key respect: The data and often the control of the data are spread out over two or more geographically separate sites. Distributed database management systems are subject to many security threats additional to those present in a centralized database management system (DBMS). The development of adequate distributed database security has been complicated by the relatively recent introduction of the object-oriented database model. This new model cannot be ignored. It has been created to address the growing complexity of the data stored in present database systems.

These days there is popularity of object oriented technology because of its features, in most of the fields people are moving to adopt this technology and looking for database management system and distributed database management system.

OODBMS allow object-oriented programmers to develop the product, store them as objects, and replicate or modify existing objects to make new objects within the OODBMS. Because the database is integrated with the programming language, the programmer can maintain consistency within one environment, in that both the OODBMS and the programming language will use the same model of representation. Relational DBMS projects, by way of contrast, maintain a clearer division between the database model and the application.

Since distributed object oriented database is new technology and not having maturity, the relative immaturity of the object-oriented model is particularly evident in distributed applications. Inconsistent standards is an example: Developers have not embraced a single set of standards for distributed object-oriented databases, while standards for relational databases are well established [9]. In the present study, security issues concerns to objected oriented database for distributed environment will be focused.

Characteristics of General database security

1. It must have physical integrity (protection from data loss caused by power failures or natural disaster),
2. It must have logical integrity (protection of the logical structure of the database),
3. It must be available when needed,
4. The system must have an audit system,
5. It must have elemental integrity (accurate data),
6. Access must be controlled to some degree depending on the sensitivity of the data,

7. A system must be in place to authenticate the users of the system, and
8. Sensitive data must be protected from inference [8].

Object-oriented Database Security

Object-oriented Databases

The basic element of an object-oriented database is the object. An object is defined by a class. Classes are arranged in a hierarchy, the root class is found at the top of the hierarchy. This is the parent class for all other classes in the model. We say that a class that is the descendent from a parent inherits the properties of the parent class. As needed, these properties can be modified and extended in the descendent class [7].

Object consist of two elements: data (variable) and methods. An object holds three basic variables types: (1) Object class: This variable keeps a record of the parent class that defines the object. (2) Object ID (OID): A record of the specific object instance. The OID is also kept in an OID table. The OID table provides a map for finding and accessing data in the object-oriented database. As we will see, this also has special significance in creating a secure database. (3) Data stores: These variables store data in much the same way that attributes store data in a relational tuple [7].

Methods are the actions that can be performed by the object and the actions that can be performed on the data stored in the object variables. Methods perform two basic functions: They communicate with other objects and they perform reads and updates on the data in the object. Methods communicate with other objects by sending messages. When a message is sent to an object, the receiving object creates a subject. Subjects execute methods; objects do not. If the subject has suitable clearance, the message will cause the subject to execute a method in the receiving object. Often, when the action at the called object ends, the subject will execute a method that sends a message to the calling object indicating that the action has ended [7].

Methods perform all reading and writing of the data in an object. For this reason, we say that the data is *encapsulated* in the object. This is one of the important differences between object-oriented and relational databases [7]. All control for access, modification, and integrity start at the object level. For example, if no method exists for updating a particular object's variable, then the value of that variable is constant. Any change in this condition must be made at the object level.

Security Issues

Access Controls

The access is controlled by classifying elements of the database. The basic element of this classification is the object. Access permission is granted if the user has sufficient security clearance to access the methods of an object. Millen and Lunt [7] describe a security model that effectively

explains the access control concepts in the object-oriented model. Their model is based on six security properties:

Property 1: (Hierarchy Property). The level of an object must dominate that of its class object.

Property 2: (Subject Level Property). The security level of a subject dominates the level of the invoking subject and it also dominates the level of the home object.

Property 3: (Object Locality Property). A subject can execute methods or read or write variables only in its home object.

Property 4**:** (*-Property) A subject may write into its home object only if its security is equal to that of the object.

Property 5**:** (Return value property) A subject can send a return value to its invoking subject only if it is at the same security level as the invoking subject.

Property 6**:** (Object creation property) The security level of a newly-created object dominates the level of the subject that requested the creation [7].

Out of above stated 6 properties, property 1 ensures that the object that inherits properties from its parent class has at least the same classification level as the parent class. If this were not enforced, then users could gain access to methods and data for which they do not have sufficient clearance. Property 2 ensures that the subject created by the receiving object has sufficient clearance to execute any action from that object. Hence, the classification level given to the subject must be equal to at least the highest level of the entities involved in the action. Property 3 enforces encapsulation. If a subject wants to access data in another object, a message must be sent to that object where a new subject will be created. Property 6 states that new objects must have at least as high a clearance level as the subject that creates the object. This property prevents the creation of a covert channel. Properties 4 and 5 are the key access controls in the model. Property 4 states that the subject must have sufficient clearance to update data in its home object. If the invoking subject does not have as high a classification as the called object's subject, an update is prohibited. Property 5 ensures that if the invoking subject from the calling object does not have sufficient clearance, the subject in the called object will not return a value. The object-oriented model and the relational model minimize the potential for inference in a similar manner. Remaining consistent with encapsulation, the classification constraints are executed as methods. If a potential inference problem exists, access to a particular object is prohibited [7].

Integrity

The integrity constraints are executed at the object level [7]. These constraints are similar to the explicit constraints used in the relational model. The difference is in execution. An object-

oriented database maintains integrity before and after an update by executing constraint checking methods on the affected objects.

Encapsulation benefit is that subjects from remote objects do not have access to a called object's data. This is a real advantage that is not present in the relational DBMS. An object oriented system derives a significant benefit to database integrity from encapsulation. This benefit stems from modularity. Since the objects are encapsulated, an object can be changed without affecting the data in another object. So, the process that contaminated one element is less likely to affect another element of the database.

Object-Oriented Database Security Problems in the Distributed Environment

No support for views in OODBs

Views plays an import role in security but OODBs do not support views. Although there have been several proposals. The development of an object-oriented view capability is complicated by such model features as object identity. What are the identities of the objects in a view? On the other hand, there has also been the argument that data encapsulation and inheritance make explicit view definitions unnecessary [10].

Security concerns with OODBs

While RDBs support authorization, most OODBs do not support authorization [11]. RDBs allow users to grant and revoke privileges to read or change the definitions and tuples in relations and views [11]. If OODBs are going to expand into more business oriented fields, this feature has to be improved.

Some OODBs require users to explicitly set and release locks. RDBs automatically set and release locks in user processing query and update statements [11].

The organization of the object-oriented DDBMS is more difficult than the relational DDBMS. In a relational DDBMS, the role of client and server is maintained. This makes the development of multilevel access controls easier. Since the roles of client and server are not well defined in the object-oriented model, control of system access and multilevel access is more difficult. System access control for the object-oriented DDBMS can be handled at the host site in a procedure similar to that described for the relational DDBMS. Since there is no clear definition of client and server, however, the use of replicated multisite approval would be impractical.

Multilevel access control problems arise when developing effective and efficient authorization algorithms for subjects that need to send messages to multiple objects across several geographically separate locations. According to Sudama [9], there are currently no universally accepted means for enforcing subject authorization in a pure object-oriented distributed environment.

This means that, while individual members have developed there own authorization systems, there is no pure object-oriented vendor-independent standard which allows object-oriented database management systems (OODBMS) from different vendors (a heterogeneous distributed system) to communicate in a secure manner. Without subject authorization, the controls described in the previous section cannot be enforced. Since inheritance allows one object to inherit the properties of its parent, the database is easily compromised. So, without effective standards, there is no way to enforce multilevel classification. Sudama [9] notes that one standard does exist, called OSF DCE (Open Software Foundation's Distributed Computing Environment), that is vendor-independent, but is not strictly an object-oriented database standard.

While it does provide subject authorization, it treats the distributed object environment as a client/server environment as is done in the relational model. He points out that this problem may be corrected in the next release of the standard.

The major integrity concern in a distributed environment that is not a concern in the centralized database is the distribution of individual objects. Recall that a RDBMS allows the fragmentation of tables across sites in the system. It is less desirable to allow the fragmentation of objects because this can violate encapsulation. For this reason, fragmentation should be explicitly prohibited with an integrity constraint.

Discussion

Since the structure and properties of OODBMS are complex then RDBMS We have mentioned OODBMS security concern in distributed environment we have to take extra efforts for maintaining OODBMS security in distributed environment we have mentioned different security concerns like we have to protect multilevel data at the object level through subject authorization and limitation of access to the object's methods. The principle unsolved problem in centralized databases is inference. The current strategies do not prevent all forms of inference and those suggested by researchers are computationally cumbersome.

There is relative immaturity security wise of the distributed object-oriented database over distributed relational database system. The relational model, however is not without problems: The processing of global views in a heterogeneous environment takes too long, and the enforcement of database integrity in a heterogeneous environment is problematic because of the conflicts between local and global integrity constraints.

The lack of completely compatible, vendor-independent standards for the distributed OODBMS relegates this model to a promised, yet not completely delivered, technology. If the distributed environment is homogeneous, the implementation of subject authorization should be

possible. For the heterogeneous distributed OODBMS, however, the absence of universally accepted standards will continue to hamper security efforts.

There are some strength but also some weakness in OODBMS in distributed environment which has been discussed in earlier before. Also some security issues which is mentioned in earlier sections we have to consider for maintaining OODBMS in distributed environment.

Conclusion

The database security issues has be discussed in distributed environment for OODBMS which are quite different than RDBMS because of its structure complexity and properties. This model have some strengths and weaknesses. Because of vendor independence of the model there is lack of security standards.

These days hybrid models are popular in market which combine the features of the two models discussed raise many new security questions. For example, Informix's Illustra combines a relational database schema with the capability to store and query complex data types. They call this system an "object-relational database." Informix claims that their system has all the capabilities of a RDBMS, including "standard security controls" with the principle advantage of an OODBMS: encapsulation, inheritance, and direct data access through the use of data IDs. This hybrid and similar systems offered by Oracle and others raise many new questions. For example,

Do the relational database security controls work well with complex data types and objects? How well do these security controls interface with encapsulation and object methods? What new avenues of attack have been opened by the combination of these two seemingly different concepts? What special security problems will arise when the object relational system is extended to the distributed environment?

Apart from these questions which are discussed above, Also there are some opportunities for research in several other areas. They include subject authorization strategies for heterogeneous distributed systems, inference prevention strategies for both centralized and distributed database systems, and distributed object-oriented database security standards.

References

An introduction to database system revised edition by Bipin Desai

[BellGris92] Bell, David and Jane Grisom, *Distributed Database Systems*. Workinham, England: Addison Wesley, 1992.

Database System concepts by Abraham Silberschatz, Henry Korth, S. Sudarshan

Distributed Database principles & system by S Tefano Ceri

[Denn87a] Denning, Dorothy E. et al., "Views for Multilevel Database Security," In *IEEE Transactions onSoftware Engineering*, vSE-13 n2, pp. 129-139, February 1987.

Fundamental of Database System Third Edition by Elmasri Navathe

[MilLun92] Millen, Jonathan K., Teresa F. Lunt, "Security for Object-oriented Database Systems," In *Proceedings IEEE Symposium on Research in Security and Privacy*, pp. 260-272,1992.

[Pflee89] Pfleeger, Charles P., (1989) *Security in Computing*. New Jersey: Prentice Hall. 1989.

[Sud95] Sudama, Ram, "Get Ready for Distributed Objects," *Datamation*, V41 n18, pp. 6771, October 1995.

Dittrich, K.A., and Dittrich, K.R., "Where Object-Oriented DBMSs Should Do Better: A Critique Based on Early Experiences," in *Modern Database Systems:The Object Model, Interoperability and Beyond*, pp. 238-252, Kim, W., ed., ACM Press, Addison Wesley, 1995.

Kim, W., "Object-Oriented Database Systems: Promises, Reality, and Future," in *Modern Database Systems: The Object Model, Interoperability and Beyond*, pp. 255-280, Kim, W., ed., ACM Press, Addison Wesley, 1995.