Security

Marta Oliva

PID_00179810

Universitat Oberta de Catalunya

The texts and images contained in this publication are subject -except where indicated to the contrary- to an Attribution-NonCommercial-NoDerivs license (BY-NC-ND) v.3.0 Spain by Creative Commons. You may copy, publically distribute and transfer them as long as the author and source are credited (FUOC. Fundación para la Universitat Oberta de Catalunya (Open University of Catalonia Foundation)), neither the work itself nor derived works may be used for commercial gain. The full terms of the license can be viewed at http://creativecommons.org/licenses/by-nc-nd/3.0/legalcode

Index

Introduction

One of the functions of database management systems (DBMS) is to provide security mechanisms to guarantee both confidentiality, integrity and availability of the data.

After identifying the potential threats to a system, this teaching module will introduce the mechanisms that can be used to ensure confidentiality, including both security controls and intrusion detection mechanisms. We will look in detail at access control security models (DAC, MAC and RBAC) and deal with the specific case of statistical databases, given the importance of techniques to protect against inference in these databases.

In the light of advances in information and communication technologies, we will address two important aspects in this area: access control in XML documents and SQL injection.

Finally, we will offer a brief introduction to current legislation on personal data protection.

Objectives

The main objective of this module is to introduce databases security. Specifically:

1. Understand the responsibilities of a database administrator from a security point of view.

2. Understand confidentiality threats and solutions available in relational DBMSs.

3. Understand the particular case of statistical databases as an example of the need for inference control.

4. Learn about some of the security issues affecting the use of new technologies.

5. Be aware of current Spanish and Catalan legislation on personal data protection.

1. Database Security Concepts

The continuous advances in information technologies have given us more tools to access information, but they have also paved the way for rising threats. Since their early appearances, DBMSs have become the main tool for storing relevant data in any organisation. It is therefore no wonder that it has now become essential to equip the DBMS with the required elements for security control.

In the field of databases, the term security encompasses three distinct concepts: confidentiality, integrity and availability. Any tool that allows us to prevent, dissuade or detect the undesired disclosure of information keeps the data **confidential**. To prevent, detect or dissuade the undesired modification of information, we ensure the **integrity** of the data. By preventing, detecting or dissuading any undesired denial of access to services, we ensure the **availability** of the information.

Note

The concepts of confidentiality, integrity and availability constitute the attributes "CIA".

Examples

- If we ensure that the employees of an organisation do not know the salaries of their managers, we are keeping these data confidential.
- By preventing employees from being able to change their salary, we ensure compliance with the data integrity of the organisation.
- When we ensure that everybody can send payment orders on time, we are guaranteeing the availability of the service.

We can guarantee data security through the various components of the DBMS. Specifically, access control mechanisms guarantee confidentiality. Data integrity is ensured by both the access control mechanism and the defined semantic integrity constraints. With this combination, whenever a subject tries to modify data, the access control mechanism will check that the user has the right to modify the data, while the semantic integrity subsystem (also known as constraint checker) will check that the updated data are semantically correct. Lastly, the recovery manager and concurrency control mechanisms ensure that the data are available and correct in spite of any hardware and software failures or concurrent accesses. This module will focus exclusively on mechanisms for ensuring data confidentiality.

Moreover, to guarantee the security of any system, it is essential for us to have an in-depth knowledge of the potential problems.

1.1. Security Issues

A **threat** is a hostile agent that, whether by chance or by the use of a specialist technique, could disclose or modify information maintained by a system, thus violating its security.

When a threat breaches the security of a system, the possible consequences are: inappropriate disclosure of information, inappropriate modification of data or denial of service. The **inappropriate disclosure** of information occurs when a user (with or without certain permissions) unduly reads data following an intentional or accidental access. This would be a clear case of violation of confidential information due to unauthorised observation that could be used to infer/deduce unauthorised information. The **inappropriate modification** of data includes all breaches of data integrity that occur as a result of incorrect handling or modification of the data. Any action that could prevent users from accessing data or using resources can lead to **denial of service**.

Threats are classed as non-fraudulent – meaning that they are unintentional or accidental – and fraudulent, also known as *intentional threats*.

Non-fraudulent threats include natural disasters or accidents such as earthquakes, floods and fires that can damage the hardware of the system and the data stored in it. These types of disaster always lead to violation of integrity or denial of service. Errors or problems with hardware or software and human errors are also classed as unintentional or accidental threats. The former can result in incorrect application of the security mechanism and, hence, unauthorised access, reading or modification of data, or denial of access to authorised users. The latter cause unintentional violations, such as incorrect data entry or the inadequate use of applications, leading to the same consequences as those of hardware and software errors.

Fraudulent threats are carried out by authorised users who abuse their privileges and authority, or by hostile agents (dishonest internal or external users) who vandalise system software and/or hardware or unduly read and write data. Examples of hostile agents include viruses, trojans and back doors. These are characterised by the legitimate use of tasks and/or applications that hide their true fraudulent aim. Specifically, a virus is a code capable of copying itself and permanently damaging the environment in which it reproduces (sometimes irreparably). A trojan is a program that looks like a utility but is actually busy gathering information to use it for fraudulent purposes. A back door is a segment of code hidden within a program that runs through a special entrance; it is thus able to avoid the protection mechanisms and can access all of the system's resources even though it does not have enough privileges to do so.

To address these potential threats, we need to spend time on both prevention and detection. We will therefore now describe some possible security controls and intrusion detection mechanisms.

1.2. Security Controls

To avoid potential threats, we can use three different types of security control: flow control, inference control and access control.

Cryptographic techniques

To guarantee the security of data in a database, cryptographic techniques as well as security controls are used.

While with **flow control** we regulate the distribution (flow) of information between accessible objects, with **inference control** we protect indirect detection of data. Moreover, **access control** ensures that all direct accesses to objects in the system only take place in accordance with the methods and rules set by the protection mechanisms.

1.2.1. Flow Control

A data flow occurs between object X and object Y when a statement reads values of X and writes values to Y.

Flow control checks that the information contained in certain objects cannot flow explicitly (via a copy) or implicitly (via groups of instructions that use intermediate objects) to objects with less protection. Flow control criteria must take into account that certain flows must be admissible, so they will need to be kept under control.

1.2.2. Inference Control

Information inference occurs when a user reads a set of data X and uses it to obtain the set of data Y (where Y = f(X)).

A pathway/inference channel is a channel through which users can locate data X and then use it to deduce Y (using Y = f(X)). The various paths/channels of inference used are indirect access, correlated data and missing data.

With indirect access, unauthorized users obtain data for which they do not have access permission through the use of conditions that are applied to data for which they really have access permission.

Examples of indirect access

1) SELECT X FROM r WHERE Y = v; (by accessing X, we can infer that there is a certain group of data that meets the condition Y=v).

2) Inserting a tuple with the same primary key as an existing tuple that is not visible to the user.

When correlated data exist (the value of data depends on the value of another data), it is easy to deduce the value of the second from the value of the first.

Example of correlated data

Z = T x K (T and K are visible, so Z can be inferred by applying the result)

Based on the absence of values in certain data, we can deduce that certain information is protected.

Example of missing data

Answers to queries that contain null values in the information that cannot be viewed, indicating that the information exists.

A further aspect related to the deduction of data is statistical inference. This type of inference is typical of "Statistical databases".

See also

Given the importance of statistical inference, we will discuss this point in section 3 on "Statistical databases".

1.2.3. Access Control

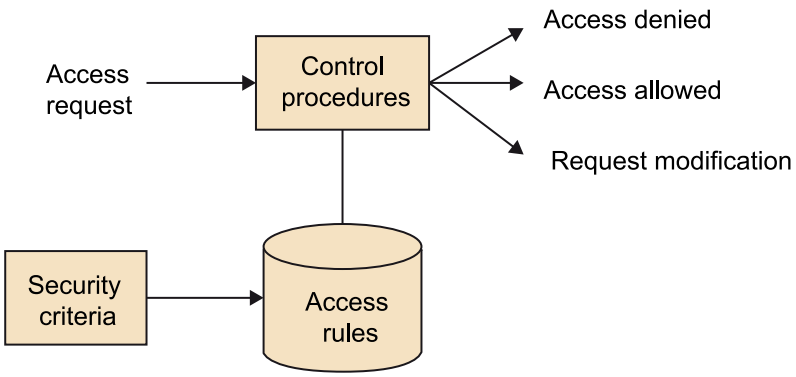
Access to the database is violated when a user (authorised or otherwise) accesses data for which he or she does not have access permissions.

The access control system is responsible for implementing access control in order to allow or deny the requested accesses. This system is based on two main sets of elements (see Figure 1):

1) A set of access rules and criteria: information stored in the system that determines the access modes to be used by the subjects in order to access objects. The subject is a system entity that initiates a request to perform an operation or series of operations on an object. Subjects can be users, processes, etc. An object is a system entity on which we can operate, such as a table or a view.

2) A set of control procedures (security mechanisms): these check the access requests against the set rules.

Figure 1. Access control system:



Source: *Database Security* (1995)

Security criteria should not be confused with security mechanisms. **Security criteria** express the basic choices made by an organisation for the security of its data, while **security mechanisms** are functions (control procedures) that implement security rules and criteria.

A number of strategies need to be decided in relation to security criteria:

- **Access restriction**: refers to how much information is accessible. We can opt for a minimum privilege criterion, which only grants authorisation for the minimum set of data that need to be known; the problem with this is that it is sometimes difficult to define the minimum set. The alternative is to use a maximum privilege criterion, which is typically used in systems that require high levels of data sharing.

- **Closed system**, in which only expressly authorised accesses are allowed, or an **open system** where, in contrast to the former, all accesses that are not explicitly denied are allowed. The choice of one or the other usually depends on whether the system requires more access permissions or more access restrictions.

- **Management of authorisations**: (who grants or revokes access rights?) if it is necessary to do this centrally, the task falls to the system security manager; if it is decentralised, it is usually the creators of the specific data who grant access rights to other users. There are also intermediate criteria: hierarchically decentralised authorisations, owner and cooperative authorisations.

- **Access control criteria**: these establish how and whether the system subjects and objects should be grouped in order to share modes of access based on the set rules and authorisations. We can choose to use criteria for discretionary access control, mandatory access control criteria (in particular, multilevel security criteria) and criteria for role-based access control.

See also

The criteria for access control will be discussed in more detail in section 2 on "Security models".

With regard to **authorisation rules**, remember that these are expressed in line with the hardware/software of the system environment that we need to protect and with the adopted security criteria.

There are internal and external security mechanisms. External mechanisms are physical and administrative control measures used to prevent unwanted access to physical resources (rooms, terminals, devices, etc.). They also include devices that protect against accidental threats. Internal mechanisms are protective measures that need to be applied to users once they have passed the external controls: authentication, access control and audit mechanisms.

User authentication is essential because we need to uniquely identify the database users. Identification is the basis of any authorisation mechanism. It involves validating the identity of the user through an object that the user has, information known to the user or a combination of the two. The systems based on information known to the user are:

- Password-based systems: where the user is authenticated by a secret string of characters known only to the user and the system.

- Question/answer systems: the user is authenticated according to the answer he or she gives to the question asked by the system. The questions generated by the system are specific to each user. Examples could include: "What city where you born in?" or "What is your favourite pet?"

- Double authentication systems (also known as handshaking): once the user is authenticated with a password, an exchange of questions begins between the system and the user based on information known only to the user (e.g. the date, time or code of the last session).

Systems based on the user having an object generally require the user to have a card that is inserted in a reader device.

Systems based on specific characteristics of the user tend to be systems that use a recorded image of the user or his/her fingerprint, or which recognise the manual signature of the user, use voice recognition or analyse the features of the user's retina.

See also

Given the importance of access control mechanisms, and considering that they must be intrinsically linked to the criteria defined by the organisation for this purpose, we will deal with this issue in more detail when we discuss "Security models" in Section 2.

Audits consist of the ability to record all accesses to data in order to analyse the sequences of access to the database. Audit mechanisms are an important element of intrusion detection and we will discuss them now.

1.3. Intrusion Detection

Intrusion detection is used to identify users intrusions by individuals using the system without authorisation (crackers) or by those who do have authorisation but are abusing their privileges (internal threats). Its purpose is to ex-

plore the different categories of security violation that can take place (data disclosure, violation of integrity, denial of service and unauthorised access) despite the security mechanisms put in place by the organisation.

For this purpose, it is necessary to maintain and review audit logs. An audit log records all requests made to the system by each user. Note, however, that the audit logs review is a complex task, takes place after the intrusion (with the necessary time delay between the security violation and the time it is detected) and is often performed manually. Since it is not convenient to do it manually or for there to be a long delay before detection, we need to be able to conduct automated audits. Thus, automated systems are useful for intrusion detection.

There are two system models for intrusion detection: **anomaly detection models** and **abuse detection models**. In the former, a normal user behaviour profile is statistically compared to the parameters of the current session; if there is any significant deviation in the behaviour, the head of the security system is notified. The second type of model compares the parameters of the current session with techniques that are commonly known to be used by attackers to penetrate the system.

Besides making audit-log processing easier, intrusion detection systems allow the detection of both offline and online security violations (in real time). The threats that can be addressed include those by external users who are not authorised to access the system, those by authorised users who make unauthorised use of system resources and those by authorised users who abuse their privileges. A series of actions is adopted to help recognise these threats. Failed logins can be used to detect attempts made by unauthorised external users. Threats from internal users can be detected by failed accesses to files or system resources or by deviations from normal patterns of behaviour with respect to system resources.

2. Security Models

The purpose of security modelling is to produce a high-level conceptual model independent of the software, based on the specifications of the requirements describing the protection needs of a system. **Security models** allow developers to provide a high-level definition of the protection requirements and system criteria and to produce a precise and concise description of the desired system behaviour.

Note

DAC stands for discretionary access control, MAC stands for mandatory access control, and RBAC stands for role-based access control.

Security models are classified according to whether they perform discretionary access control, mandatory access control or role-based access control.

2.1. Discretionary Access Control

In discretionary access control, a subject's access to information is based on the identity of that subject and the rules specifying, for each subject and object of the system, the type of access to the object permitted for the subject. The subject access request is checked against the specified authorisations. The administration of authorisations can be centralised or decentralised although the general scenario is that one user is able to grant access rights to other users. This mechanism provides a flexible way of enforcing different protection requirements.

The main DAC model is the Access Matrix Model, in which authorisations are stored in a matrix where the subjects and objects are correlated with the authorisations available to each subject in relation to each object.

Figure 2. Permissions matrix

--	O_1	O_2	O_3	O_4	O_5	--	O_m
S_1	$A[S_1, O_1]$	$A[S_1, O_2]$	$A[S_1, O_3]$	$A[S_1, O_4]$	$A[S_1, O_5]$	--	$A[S_1, O_m]$
S_2	$A[S_2, O_1]$	$A[S_2, O_2]$	$A[S_2, O_3]$	$A[S_2, O_4]$	$A[S_2, O_5]$	--	$A[S_2, O_m]$
S_3	$A[S_3, O_1]$	$A[S_3, O_2]$	$A[S_3, O_3]$	$A[S_3, O_4]$	$A[S_3, O_5]$	--	$A[S_3, O_m]$
--	--	--	--	--	--	--	--
S_n	$A[S_n, O_1]$	$A[S_n, O_2]$	$A[S_n, O_3]$	$A[S_n, O_4]$	$A[S_n, O_5]$	--	$A[S_n, O_m]$

Source: Database Security (1995)

Figure 2 contains an example of an authorisation matrix, where A[s,o] indicates the means by which *s* is authorised to access *o*, and the condition that must be met in order to allow access. DBMSs like Oracle store the authorisation matrix in the data dictionary but it can be stored in different ways: by rows, columns, or cells.

The conditions included in the authorisations can be:

- Data dependent. For example, the user can only read data on employees with salaries of <6000.
- Time dependent. For example, the user can only read data on employees between 8 am and 5 pm.
- Context dependent. For example, the user can read data on the names and salaries of employees but cannot do so at the same time or make the connection between the two.
- History dependent. For example, the user can only read employees salaries if he or she has not yet read their names.

Recommended website

With Oracle Virtual Private Database (VPD), we can dynamically add constraint conditions to the WHERE clause of SQL statements.
http://docs.oracle.com/cd/B28359_01/server.111/b28337/tdpsg_securing_data.htm

The administration of authorisations in the Access Matrix Model is performed through a decentralised system. In a decentralised system, it is the owner of the object (usually the creator) who grants and/or revokes subject access rights. Certain individuals can also be given authorisation to grant and/or revoke access rights on objects of which they are not the owners. To guarantee compliance with authorisations at all times, given the possibility that authorisation could have been granted to a subject for whom authorisation to authorise has just been withdrawn, application of the revocation of authorisations algorithm (Griffiths-Wade Algorithm) is required. This algorithm takes into account the moment at which each authorisation takes place. When an authorisation A_1 is revoked (which had been granted at time T_1) to a user U, the system needs to act as though the user U had never received the authorisation: we must therefore revoke in cascade any authorisations that the user U may have granted through the authorisation A_1. If the user U received a second authorisation A_2 (of the same type) at time T_2, then the authorisations granted to the user U from time T_2 can be maintained. Thus, it is simply a matter of revoking the authorisations made between T_1 and T_2.

In standard SQL, the clauses used to grant or revoke authorisations are `GRANT` and `REVOKE`. The syntax of these clauses is as follows:

```
GRANT <privileges> ON <objects> TO <subjects> [WITH GRANT OPTION];
REVOKE [GRANT OPTION FOR] <privileges> FROM <subjects>
 [RESTRICT | CASCADE];
```

Recommended website

A new standard SQL has just been released, SQL:2011; ISO/IEC 9075:2011. For more information, visit: http://fr.wikipedia.org/wiki/Structured_Query_Language

Here, <privileges> specifies the list of privileges (there can also be just one) to grant/revoke, <objects> is the list of objects (there can also be only one) on which we wish to grant privileges and <subjects> is the list of subjects (there can also be just one) for whom we wish to grant/revoke privileges. The `WITH GRANT OPTION` clause is used to grant the privilege of granting privileges.

The privileges that can be granted for relations (or tables) include:

- `INSERT`: for inserting tuples in the indicated relations.
- `UPDATE`: for updating tuples in the indicated relations.
- `DELETE`: for deleting tuples from the indicated relations.
- `SELECT`: for querying the information on the indicated relations.
- `REFERENCES`: for creating a foreign key.
- `TRIGGER`: for creating a trigger in the specified table.
- `EXECUTE`: for executing the specified function or procedure.
- `ALL PRIVILEGES`: for granting all privileges.

Example

Let's take the example of a database with the schema:

```
library(code, address, city, telephone, capacity)
book(ISBN, title, publisher, publication_year)
person(ID, name, title, address, city, birth_year)
author(ISBN, ID)
      FOREIGN KEY (ISBN) REFERENCES book(ISBN),
      FOREIGN KEY (ID) REFERENCES person(ID),
has(code, ISBN, quantity)
      FOREIGN KEY (code) REFERENCES library(code),
      FOREIGN KEY (ISBN) REFERENCES book(ISBN),

GRANT INSERT, DELETE, UPDATE ON library, book, has TO library_admin;
GRANT ALL PRIVILEGES ON has TO Joan;
GRANT ALL PRIVILEGES ON library TO library_super_admin WITH GRANT OPTION;
REVOKE SELECT ON has TO Joan;
```

Note

When expressing a database schema, we use underscore to indicate primary keys and italics to indicate foreign keys.

With the three `GRANT` statements, we can grant different privileges to different users of the database. Moreover, use of the `WITH GRANT OPTION` in the third `GRANT` statement also grants the privilege of granting privileges to `library_super_admin`. In the `REVOKE` statement, one of the privileges previously granted to the user Joan is revoked.

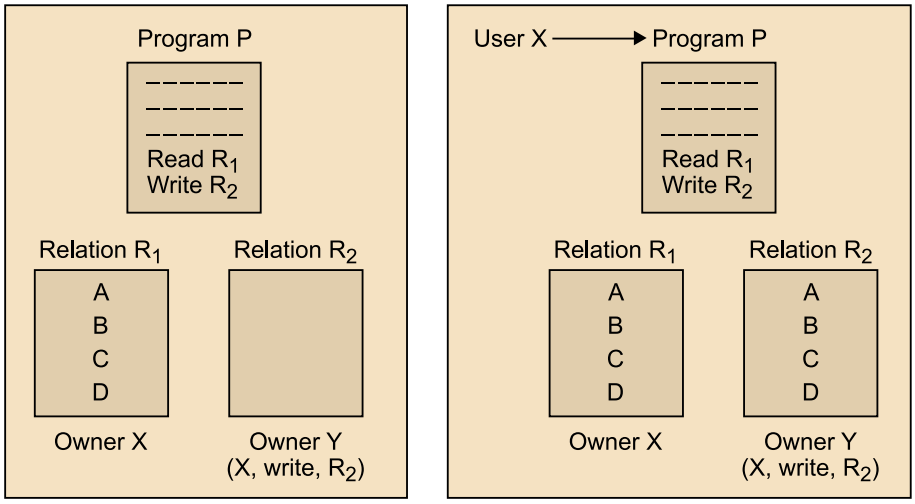
Nonetheless, discretionary access control has the following disadvantages:

- Discretionary criteria do not impose restrictions on how the information is used once it has been obtained by the user.
- The information disclosure is not controlled (the user who has obtained the data could pass them on to somebody else).
- Vulnerability to trojans.

Figure 3 shows an example of a trojan. Specifically, the program P contains two lines of hidden code (Read R_1 and Write R_2). The user Y does not have permission to execute the program P or to read the relation R_1, but does have permission to read the relation R_2. Due to the permissions that X has to exe-

cute the program P and to read R_1 and write R_2 during execution of the program P, the data are copied from R_1 to R_2 (by running the two lines of hidden code). Hence, there is a flow of information allowing the user Y to obtain data from the relation R_1 through authorised access to R_2.

Figure 3. Example of a trojan



Source: *Database Security* (1995)

The development of mandatory access criteria (MAC) was necessary to provide protection against trojans.

2.2. Mandatory Access Control

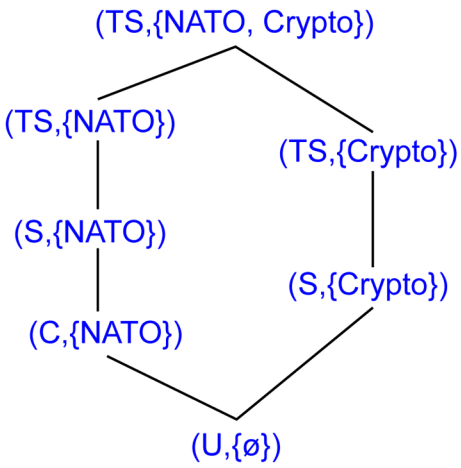
With mandatory access control, access to information by subjects is based on the classification of the subjects and objects of the system. A subject can access an object if there is some sort of relationship, depending on the mode of access, between the classification of the subject and the object.

The model of Bell & La Padula (1975) is usually adopted when MAC is applied. This is a development of the Access Matrix Model in which system elements are classified. The classification is expressed in terms of security levels. Each subject is assigned to a security level, known as a **clearance level**, indicating the level of trust that he or she has. Each cleared user at a certain level can set up a session to work at his or her clearance level or at any level below it. Each object is assigned a security level, which indicates the degree of confidentiality of the object.

Bibliography

Bell, D. E.; La Padula, L. J. (1975). *Secure Computer Systems: Unified Exposition and Multics Interpretation*. MITRE Corp., Bedford, MA.

Each security level is defined by two components: the classification and the set of categories. Typically, the classification is one of a series of four elements: Top Secret (TS), Secret (S), Confidential (C) and Unclassified (U). This set is fully ordered: TS > S > C > U. The set of categories is a subset of a set of elements without a hierarchy (e.g. NATO, Nuclear, Crypto, etc.). The set of security levels form a network, or lattice, which is partially ordered according to the dominate relationship (≥).



Given $L_1 = (C_1, S_1)$ and $L_2 = (C_2, S_2)$
$L_1 \geq L_2$ (L_1 dominates L_2) $\Leftrightarrow C_1 \geq C_2 \wedge S_1 \supseteq S_2$
$L_1 > L_2$ (L_1 strictly dominates L_2) $\Leftrightarrow C_1 > C_2 \wedge S_1 \supset S_2$
$L_1 \leq L_2$ (L_1 is dominated by L_2) $\Leftrightarrow C_1 \leq C_2 \wedge S_1 \subseteq S_2$
$L_1 < L_2$ (L_1 is strictly dominates L_2) $\Leftrightarrow C_1 < C_2 \wedge S_1 \subset S_2$
$\neg(L_1 \geq L_2) \wedge \neg(L_1 \leq L_2) \Rightarrow L_1$ and L_2 are incomparable.

The confidentiality of the data is expressed by a set of rules (axioms) that must be met during system operation. This model is governed mainly by the following basic axioms:

1) Simple security property

A subject can only read an object if his or her clearance level dominates the security level of the object (No read-up secrecy).

2) Star (*) Property

A subject can only write an object if the security level of the object dominates the subject's clearance level (No write-down secrecy).

The security classifications introduced by Bell & La Padula and Jajodia & Sandhu (1991) defined a model for the application of mandatory criteria in relational DBMS. It is a development of the relational model in that it considers security classifications in which multilevel relations have two parts: a multilevel relation schema, which is independent of the current database state, and a collection of tuples that do depend on the database state.

Bibliography

Jajodia, S.; Sandhu, R. (1991). "Towards a multilevel relational data model". In proceedings of ACM-SIGMOD Conference, Denver, CO.

Figure 4. Example of a multilevel relation

Multilevel Employee relation

Name	C_{Name}	Dept.	C_{Dept}	Salary	C_{Salary}	TC
Bernat	S	Dept1	S	10.000	S	S
Anna	S	Dept2	S	20.000	TS	TS
Sara	TS	Dept2	TS	30.000	TS	TS

Instance at level S

Name	C_{Name}	Dept.	C_{Dept}	Salary	C_{Salary}	TC
Bernat	S	Dept1	S	10.000	S	S
Anna	S	Dept2	S	–	S	S

Instance at level TS

Name	C_{Name}	Dept.	C_{Dept}	Salary	C_{Salary}	TC
Bernat	S	Dept1	S	10.000	S	S
Anna	S	Dept2	S	20.000	TS	TS
Sara	TS	Dept2	TS	30.000	TS	TS

Figure 4 shows an example of a multilevel relation, the Employee multi-level relation. The header of this relation includes an additional attribute for the classification of each attribute (C_{Name}, C_{Dept}, C_{Salary}) and an attribute for tuple classification (TC). For each tuple, it also specifies the classification of the value of each attribute (not always available in commercial versions of DBMSs) and the global classification of the tuple. In addition, it shows the partitions of the relation, where each partition corresponds to a different security level. Thus, a subject with security level *c* can read all the tuples in the partitions corresponding to security levels *c* and below. By contrast, the subject can write to the partitions whose classification is *c* or above.

Figure 5. Example of updates to the Employee multilevel relation in Figure 4.

a) User S: `INSERT INTO Employee VALUES ('Joan', 'Dept2', 20.000);`

Instance at level S of multilevel Employee relation

Name	C_{Name}	Dept.	C_{Dept}	Salary	C_{Salary}	TC
Bernat	S	Dept1	S	10.000	S	S
Anna	S	Dept2	S	–	S	S
Joan	S	Dept2	S	20.000	S	S

Instance at level TS of multilevel Employee relation

Name	C_{Name}	Dept.	C_{Dept}	Salary	C_{Salary}	TC
Bernat	S	Dept1	S	10.000	S	S
Anna	S	Dept2	S	20.000	TS	TS
Sara	TS	Dept2	TS	30.000	TS	TS
Joan	S	Dept2	S	20.000	S	S

b) User TS: `INSERT INTO Employee VALUES ('Bernat', 'Dept2', 20.000);`

Instance at level S of multilevel Employee relation

Name	C_{Name}	Dept.	C_{Dept}	Salary	C_{Salary}	TC
Bernat	S	Dept1	S	10.000	S	S
Anna	S	Dept2	S	–	S	S
Joan	S	Dept2	S	20.000	S	S

Instance at level TS of multilevel Employee relation

Name	C_{Name}	Dept.	C_{Dept}	Salary	C_{Salary}	TC
Bernat	S	Dept1	S	10.000	S	S
Bernat	TS	Dept2	TS	20.000	TS	TS
Anna	S	Dept2	S	20.000	TS	TS
Sara	TS	Dept2	TS	30.000	TS	TS
Joan	S	Dept2	S	20.000	S	S

c) User S: `DELETE FROM Employee WHERE name = 'Anna';`

Instance at level S of multilevel Employee relation

Name	C_{Name}	Dept.	C_{Dept}	Salary	C_{Salary}	TC
Bernat	S	Dept1	S	10.000	S	S
Anna	**S**	**Dept2**	**S**	**–**	**S**	**S**

Instance at level TS of multilevel Employee relation

Name	C_{Name}	Dept.	C_{Dept}	Salary	C_{Salary}	TC
Bernat	S	Dept1	S	10.000	S	S
Bernat	S	Dept2	TS	20.000	TS	TS
Anna	**S**	**Dept2**	**S**	**–**	**S**	**S**
Anna	S	Dept1	TS	30.000	TS	TS
Sara	TS	Dept2	TS	30.000	TS	TS

Figure 5 a) shows the result of an insertion in the Employee relation made by a level S user. The insertion affects both the partition corresponding to the instance of the relation classified as level S and the partition corresponding to the instance of the relation classified as level TS. As with the deletions in Figure 5 c), the tuples in red will be deleted. By contrast, in the insertion in Figure 5 b), the insertion made by a TS-level user will only affect the TS partition.

In certain implementations of the multilevel relational model, write operations affecting security levels higher than those of the subject are not permitted for integrity reasons. The fact that a single tuple may contain attributes classified into different levels means that the tuple can belong to several partitions of a multi-level relation (such as the tuple for Anna in Figure 4), resulting in polyinstantiation of the tuple and, hence, possible updating difficulties. To manage polyinstantiation, we must revise several classical concepts of the relational model, such as the notion of the primary key. Due to these difficulties, commercial implementations of the multilevel relational model only support tuples whose attributes are all classified at the same level.

Polyinstantiation

The term polyinstantiation is used to denote the existence of two or more tuples with the same primary key value that are classified into different security levels.

Note that in the classical relational model, the definition of the primary key is based on functional dependencies: the value of the attributes forming the primary key determines the value of all other attributes. As a result, no two tuples in any relation can have the same values for the attributes that form the primary key. By contrast, in multilevel relations, in order to prevent users from deducing information for which they do not have authorisation, a distinction is made between the *apparent primary key* (AK) and the *actual primary key*. The AK is the primary key defined in the relational schema. The actual primary key is defined from the property of *polyinstantiation integrity*.

A multilevel relation R only satisfies the polyinstantiation integrity if, based on the apparent primary key, its classification and the classification of an attribute, we can functionally determine the value of this attribute. Thus, the actual primary key must be defined as the union of the apparent primary key, the classification of the primary key and the set of classifications of all non-key attributes.

$AK \cup C_{AK} \cup C_R \rightarrow A_R$

where C_R is the set of classification attributes of non-key attributes and A_R is the set of all non-key attributes.

If we look back at the Employee multilevel relation in Figure 4, with the apparent key *Name*, the classification of this key C_{Name} and the classification of the *Salary* attribute C_{Salary}, we can determine the value of *Salary*.

Example: Anna $\cup$ S $\cup$ S $\rightarrow$ null, whereas Anna $\cup$ S $\cup$ TS $\rightarrow$ 20,000.

As we saw with DAC, MAC also has disadvantages:

- The mandatory criteria are too rigid and irrelevant in certain environments. Specifically, it is not always possible to assign clearance levels to subjects and/or security levels to objects in commercial information systems.

- Security criteria are required, which must be both discretionary and mandatory. This allows the security system to protect itself from trojans, since it is possible to control the information flow.

2.3. Role-based Access Control

Neither discretionary nor mandatory criteria meet the needs of most businesses. Criteria are needed to specify the authorisations granted to users (or groups) on objects (as in DAC), as well as the possibility of specifying constraints (as in DAC) for allocating or using these authorisations. These needs led to the use of role-based (RBAC) criteria, as a direct development of discretionary criteria.

Bibliography

The first standard for RBAC was published in 2001:

Ferraiolo, D. F.; Sandhu, R. et al. (2001). "Proposed NIST Standard for Role-Based Access Control". *ACM Transactions on Information and System Security* (Vol. 4, No. 3, August, p. 224–274).

The term *role* is used to describe the set of actions and responsibilities associated with a given working activity.

With RBAC criteria, we regulate access to information by subjects based on the activities that they execute on the system. Roles need to be identified in the system. Hence, authorisations for access on objects are specified by roles. Users then assume the authorisations of the role they adopt.

In standard SQL, there are clauses for working with roles[1]. The main ones are `CREATE ROLE` and `DROP ROLE`, for creating and destroying roles, respectively. Executing `CREATE ROLE` clause creates a set of privileges that can be assigned to the database users. When we assign a role to a user, he or she receives all the privileges of that role. The set of privileges assigned to a role can be modified by granting or revoking privileges using the `GRANT` and `REVOKE` clauses. To allow users to make use of the privileges assigned to a role, the user has to activate the role, which is done using the `SET ROLE` clause. The syntax for all role-related SQL clauses is:

(1)Roles were incorporated into the standard SQL in 1999, in SQL:1999.

```
CREATE ROLE <name> [ WITH ADMIN rolename ];
DROP ROLE <name>;
SET ROLE <name>;
```

where <name> is the name used to identify the role. The optional clause `WITH ADMIN` is used to define which role, with name <rolename>, has the right to grant membership of this role to others.

Example

For this example, we will use the same database schema as we used in the example in 2.1 (Discretionary access control).

```
CREATE ROLE manager;
GRANT ALL PRIVILEGES ON has TO manager;
GRANT manager TO Anna;
SET ROLE manager;
REVOKE manager FROM Anna;
REVOKE ALL PRIVILEGES FROM manager;
DROP ROLE manager;
```

After executing the first three statements, the user Anna has the privileges granted to anybody with the Manager role. Thus, Anna can activate the role using `SET ROLE` when she needs to. Moreover, the user with sufficient permissions to execute the first three statements can also execute the last three and thus revoke the privileges previously granted and delete the role created.

3. Statistical Databases

A statistical database (SDB) is a database used to query statistics (such as counts, averages, sums, etc.) on a subset of occurrences in the database.

We can distinguish between two types of statistical databases: special purpose and general purpose. SDBs with a **special purpose** are only used for statistical calculations (e.g. census SDBs), while **general-purpose** SDBs are ordinary DBs (such as those of hospitals, banks, retail establishments, schools, etc.) that can be used both to access individual instances through common applications/queries (serving in this case as the security mechanisms we saw above) and to access statistics through statistics applications. In the latter case, special protection techniques are required.

SDB protection consists of guaranteeing the right to access statistical information while maintaining the confidentiality of the individual information stored in the database. Protecting a SDB also entails preventing and avoiding statistical inference.

Inference in a SDB is the ability to obtain confidential information on an occurrence from a sequence of statistical queries performed on a set of occurrences in the DB.

The first security measure is to construct a **statistical filter** that only allows statistical queries and denies direct access to specific occurrences of the SDB.

Example

A query to count how many employees earn a salary above a certain amount would be allowed but access to specific employees would be denied.

However, the use of a filter is not enough.

Example

First of all, we query the average salary of female employees and we then count the number of female employees. If there is only one female employee, we can legitimately find out her salary (using only statistical queries).

In the above example, the SDB is **positively compromised** because the user can discover that an individual meets a certain characteristic. By contrast, a SDB is **negatively compromised** if the user is able to discover that an individual does not meet a particular characteristic.

To protect a SDB, we need to consider the following information:

1) Characteristics of the SDB to protect:

- On-line (the users wait for the response to the statistical queries in real-time) or off-line (the users do not know when the statistics will be processed).
- Static (the SDB data do not vary, as in a census SDB for example) or dynamic (the data undergo changes reflecting those that occur in the real world).
- Centralised or distributed.
- Oriented towards a single individual application or a series of heterogeneous applications (the heterogeneity makes protection more complex).

2) Additional knowledge of users, which is information already known to the user and not necessarily obtained from the system but which can be used to infer other information.

3) Types of attack (complexity/sophistication).

3.1. Inference Protection Techniques

Techniques for protection against inference can be classified into three types: conceptual, perturbation-based, and restriction-based.

3.1.1. Conceptual Techniques

Conceptual techniques are so-called because they tackle the problem at a conceptual level by defining a model of inference protection for SDBs. The lattice model (Denning and Schlörer, 1983) and conceptual partitioning (Chin and Ozsoyoglu, 1981) are examples of conceptual techniques.

Bibliography

Dening, D. E.; Schlörer, J. (1983). "Inference controls for statistical databases". *IEEE Computer*, (16(7), July).

Chin, F. Y.; Ozsoyoglu, G. (1981). "Statistical database design". *ACM Trans. Database Systems*, (6(1), March).

The lattice model

This model adopts a conceptual representation of the records stored in the SDB, based on the lattice structure of m-dimensional tables (m-tables) organised into different levels of abstraction (see the example in Figure 6).

Figure 6. Cube or three-dimensional table

DSC table				
Date_of_Birth	**Gender**	**Dept_code**		
		Dept1	Dept2	Dept3
1961-1971	M	10	12	0
	F	1	0	3
1972-1982	M	12	10	5

A cube or three-dimensional table indicates the count performed on the following attributes/dimensions: Date_of_Birth, Gender and Dept_code of the Employee relation in the SDB.

DSC table				
Date_of_Birth	**Gender**	**Dept_code**		
	F	20	2	8
>1982	M	15	0	1
	F	20	10	0

A cube or three-dimensional table indicates the count performed on the following attributes/dimensions: Date_of_Birth, Gender and Dept_code of the Employee relation in the SDB.

The set of m-tables corresponding to a particular statistic will yield a lattice structure. This lattice is constructed using the aggregation (sum) mechanism for an attribute, which produces smaller tables than m until we obtain a table of 0 dimensions that represents the statistic calculated (T_{all}). Figure 7 shows the various tables produced by aggregating in each dimension.

Figure 7.

DS Table		
Date_of_Birth	Gender	
	M	F
1961-1971	22	4
1972-1982	27	30
>1982	16	30

DC Table			
Date_of_Birth	Dept_code		
	Dept1	Dept2	Dept3
1961-1971	11	12	3
1972-1982	32	12	13
>1982	35	10	1

SC Table			
Gender	Dept_code		
	Dept1	Dept2	Dept3
M	37	22	6
F	41	12	11

From the three-dimensional *DSC* table in Figure 6, we obtain the two-dimensional *DS*, *DC*, *SC* tables, by aggregation, for the count statistic.

Continuing with the tables in Figure 7, we obtain tables smaller than *m* until we reach the 0-dimensional table that represents the count on the entire SDB (see Figure 8).

Figure 8. Final result of count

Alltable
129

Figure 9. Lattice corresponding to count.

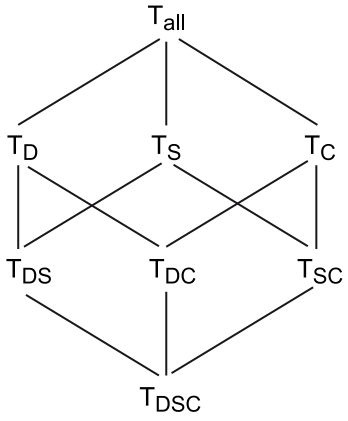


Figure 9 shows the lattice structure corresponding to the count from the three-dimensional table Date_of_Birth, Gender and Dept_code (T_{DSC}) in Figure 6.

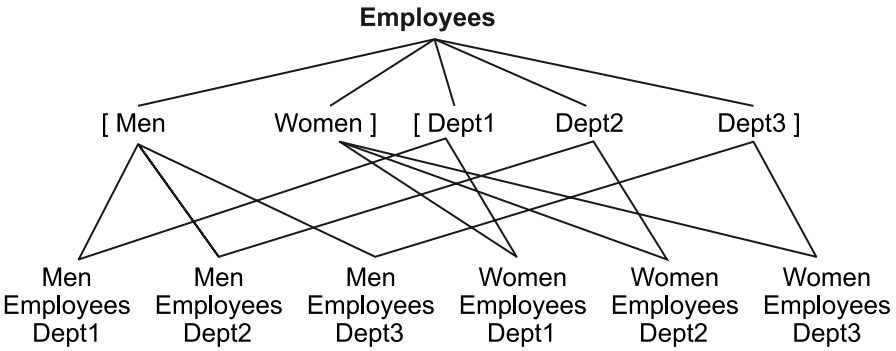
Based on the tabular form of the lattice model, different inference control techniques can be studied (of those based on restrictions or perturbations) in order to restrict sensitive statistics, considering that a statistic is sensitive when it allows confidential data about an individual to be obtained. Similarly, in an m-table, cells with the value 1 for counts are sensitive.

Conceptual partitioning

This technique is based on the conceptual definition of a set of entities in the SDB called populations on which statistics can be released, and on the conditions that must be verified to avoid inference. A population is a set of entities in the SDB with common properties.

The data abstraction model, based on the aggregation and generalisation abstractions, is used to model the set of SDB entities. With the use of generalisation hierarchies, it is possible to define different levels of generic objects by distributing the individuals according to the values of some of their properties. Thus, populations are broken down into sub-populations down to atomic sub-populations (which are the leaves of the hierarchy that cannot be broken down further). Sub-populations that have a common parent in the hierarchy form a cluster (see Figure 10).

Figure 10. Conceptual model



Like the lattice model, conceptual partitioning also provides a good framework for understanding and investigating the security issues of SDBs.

3.1.2. Perturbation or Noise-Based Techniques

These techniques consist of introducing small modifications (noise) during the processing of the statistical query. In contrast to restriction-based techniques, any query can be made but the result is only an approximation. There are two ways to introduce noise: by modifying records in the SDB (record-based perturbation techniques) or by modifying the result before sending it back to the user (result-based perturbation techniques). In both cases, the con-

sistency of the possible responses must be monitored. Rounding, query-based perturbation, fixed perturbation, random-sample queries and data swapping are all perturbation-based techniques.

The term "rounding" stems from the fact that the values of the response are rounded before being delivered to the user. Hence, it is regarded as a technique based on perturbations in results. There are three types of rounding: systematic rounding, random rounding, controlled rounding.

With systematic rounding, the result of the queried statistic is rounded to the nearest integer, which will be a multiple of a given base (for example, base 5). Figure 11 a) uses table DC containing the count results to illustrate the application of systematic rounding, base 5. Note that there are some inconsistencies. For example, the sum of the values of the Dept1 department column is 75 while the true value of the summation is 80.

In random rounding, we randomly round the result of the queried statistics to the integer immediately above or below the two multiples of the rounding base. The rounding is decided on the basis of the query, so we can always answer the same query in the same way. Figure 11 b) shows the random rounding to base 5 applied to DC table in the example.

Lastly, controlled rounding is merely a slight variation to ensure that inconsistencies do not occur, forcing the marginal sums of the rounded statistics to match their rounded sum. Figure 11 c) shows controlled rounding based on the systematic rounding in Figure 11 a).

Figure 11. Example of a) systematic rounding, b) random rounding and c) controlled rounding

DC Table				
Date_of_Birth	Dept_code			
	Dept1	Dept2	Dept3	Σ
1961-1971	11	12	3	26
1972-1982	32	12	13	57
>1982	35	10	1	46
Σ	78	34	17	129

Original values of the count on DC Table in Figure 6.

a) Values after applying systematic rounding, base 5.

DC Table				
Date_of_Birth	Dept_code			
	Dept1	Dept2	Dept3	Σ
1961-1971	10	10	5	25
1972-1982	30	10	15	55
>1982	35	10	0	45
Σ	80	35	15	130

b) Values after applying random rounding, base 5.

DC Table				
Date_of_Birth	Dept_code			
	Dept1	Dept2	Dept3	Σ
1961-1971	10	15	5	25
1972-1982	35	10	15	60
>1982	35	10	0	45
Σ	80	35	20	130

c) Values after applying controlled rounding, based on the systematic rounding obtained in a).

DC Table				
Date_of_Birth	Dept_code			
	Dept1	Dept2	Dept3	Σ
1961-1971	10	10	5	25
1972-1982	30	10	15	55
>1982	35	10	0	45
Σ	75	30	20	125

The fixed perturbation and query-based perturbation techniques modify the values of the attributes used to calculate the statistics, which are calculated using the perturbed values. With the first type, the perturbation does not vary from query to query, while in the second it does. With the random-sample queries technique, only a subset of all the results meeting the conditions of the query are shown. An important point in relation to this technique is that we must ensure that the same response is always given to the same query. Lastly, data swapping is a perturbation technique whereby attribute values are exchanged in twos among the various tuples of the original SDB, such that the resulting modified SDB has no records in common with the original SDB. This also guarantees inference protection and the correction of the statistics obtained.

3.1.3. Restriction-Based Techniques

Restriction-based techniques restrict the statistical queries that could disclose confidential information to the user about unique individuals represented in the SDB (such as those performed on very small sets of occurrences). The drawback to this type of technique is that it sometimes limits the use of the SDB since only a subset of the statistical queries can be performed.

Note

There are different types of restriction-based techniques: query-set-size control, query-set-overlap control and audit-based control.

4. Security for Advanced Data Management Systems

Continuous technological advances require security aspects to be considered in new technologies too. The extensive use of XML has made it necessary to put in place mechanisms to control access to XML documents or parts of these.

Likewise, the proliferation of web applications has increased the range of threats to database systems serving data to these applications. One of the biggest threats posed are SQL injections. We will deal with both aspects in the following two subsections.

4.1. Access Control for XML

XML is widely used nowadays because it has become the standard for describing data and documents on the Internet. XML is characterised by the use of semantic tags to mark the different data elements. Document Type Definition (DTD) and XMLSchema are also frequently used to specify document structures in a very similar way to how database schemas are specified. However, note that a document does not have to be associated with a DTD or XMLSchema.

Initially, security mechanisms were based on encrypting the XML document, meaning that whoever had a valid encryption key could access the entire document.

In recent years, efforts have been made to address security issues in XML to control access to certain parts of a document. According to Damiani et al. (2008), the proposed access control models for XML can be grouped into two broad categories: node filtering and query rewriting systems. The first of these categories includes proposals that use access policies to compute secure user views on sets of XML data. Thus, user queries are evaluated on these views. The second category of proposals uses authorisation rules to transform non-secure user requests into secure requests for their evaluation on the set of XML data.

Bibliography

Damiani, E.; Fansi, M.; Gabillon, A.; Marrara, S. (2008). "A general approach to securely querying XML". *Computer Standards & Interfaces* (Volume 30, Issue 6, p. 379–389).

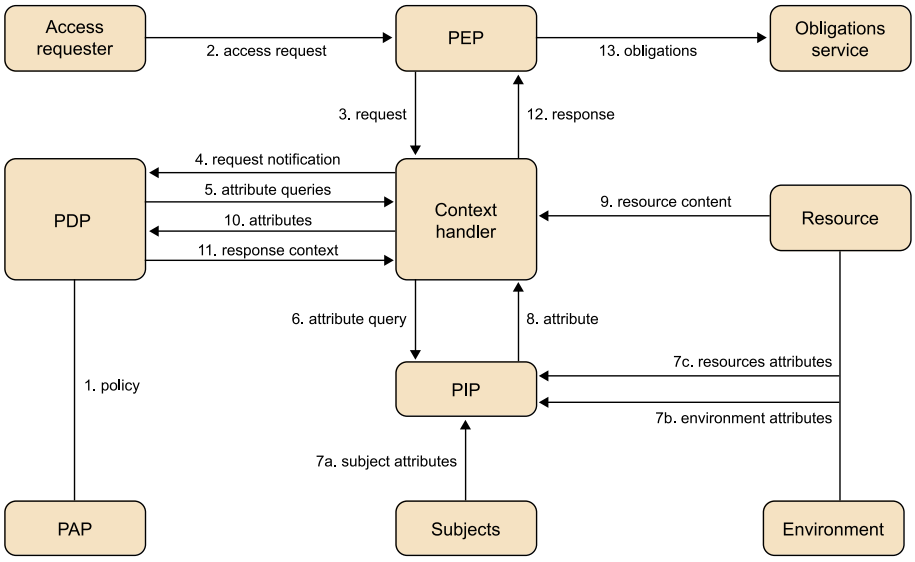
Along the same lines, in 2003 the OASIS Consortium (Organization for the Advancement of Structured Information Standards) approved a standard for access control in the form of the eXtensible Access Control Markup Language (XACML 1.0), now in its third version (XACML 3.0, August 2010). The document http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.html contains the full specification of this standard but an extract of the main points of the document now follows.

The XACML standard has a data flow model and a language model for expressing access control criteria.

The major actors in the XACML domain are shown in the data-flow diagram of Figure 12. These are:

- **Policy administration point** (PAP), the system entity that creates a **policy** or **policy set.**
- **Policy decision point** (PDP), the system entity that evaluates the **applicable policy** and renders an **authorization decision**.
- **Policy enforcement point** (PEP), the system entity that performs **access control**, by making **decision requests** and enforcing **authorization decisions**.
- **Policy information point** (PIP), the system entity that acts as a source of **attribute** values.
- **Context handler**, the system entity that converts **decision requests** in the native request format to the XACML canonical form and converts **authorization decisions** in the XACML canonical form to the native response format.
- **Environment**, the set of **attributes** that are relevant to an **authorization decision** and are independent of a particular **subject**, **resource** or **action.**
- **Resource**, data, service or system component.
- **Subject**, an actor whose **attributes** may be referenced by a **predicate** (a statement about **attributes** whose truth can be evaluated).
- **Obligation**, an operation specified in a **rule**, **policy** or **policy set** that should be performed by the **PEP** in conjunction with the enforcement of an **authorization decision.**

Figure 12. Data-flow diagram (some of the data flows may be facilitated by a repository)



The operation of the model works as follows.

1) **PAPs** write **policies** and **policy sets**, and make them available to the **PDP**. These **policies** or **policy sets** represent the complete **policy** for a specified **target**.

2) The **access** requester sends a request for **access** to the **PEP**.

3) The **PEP** sends the request for **access** to the **context handler** in its native request format, optionally including **attributes** of the **subjects**, **resource**, **action**, **environment** and other categories.

4) The **context handler** constructs an XACML request **context** and sends it to the **PDP**.

Context is the canonical representation of a **decision request** and an authorization decision.

5) The **PDP** requests any additional **subject**, **resource**, **action**, **environment** and other categories (not shown) **attributes** from the **context handler**.

6) The **context handler** requests the **attributes** from a **PIP**.

7) The **PIP** obtains the requested **attributes**.

8) The **PIP** returns the requested **attributes** to the **context handler**.

9) Optionally, the **context handler** includes the **resource** in the **context**.

10) The **context handler** sends the requested **attributes** and (optionally) the **resource** to the **PDP**. The **PDP** evaluates the **policy**.

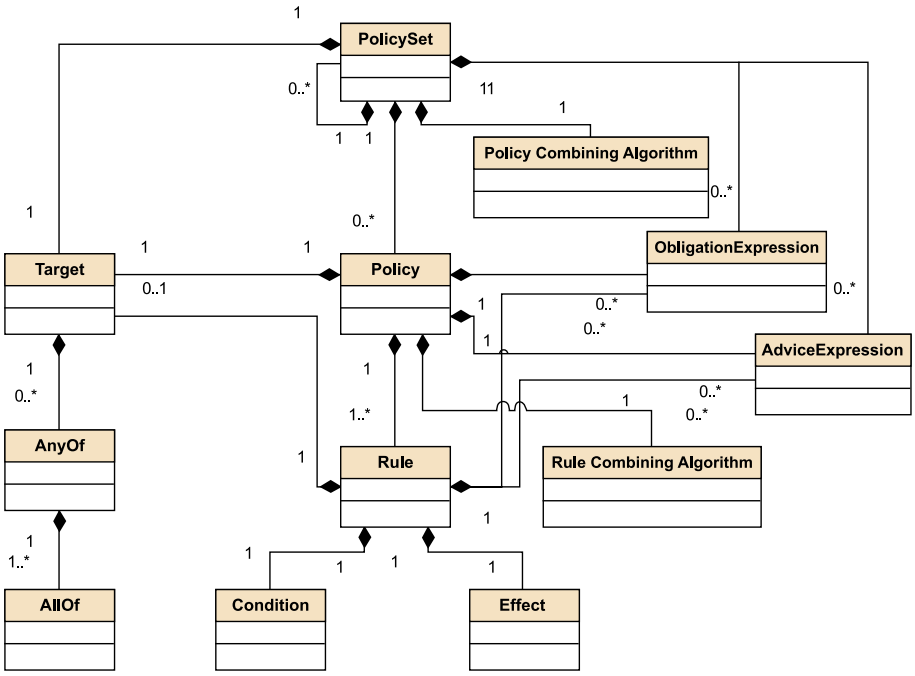
11) The **PDP** returns the response **context** (including the **authorization decision**) to the **context handler**.

12) The **context handler** translates the response **context** to the native response format of the **PEP**. The **context handler** returns the response to the **PEP**.

13) The **PEP** fulfils the **obligations**.

14) (Not shown) If **access** is permitted, then the **PEP** permits **access** to the **resource**; otherwise, it denies **access**.

Figure 13. Policy language model



The **policy** language model is shown in Figure 13. The main components of the model are:

- **Rule**, is the most elementary unit of **policy**. In order to exchange **rules** between major actors, they must be encapsulated in a **policy**. A **rule** can be evaluated on the basis of its contents. The main components of a **rule** are: a **target**, an **effect**, a **condition** (optional), **obligation expressions** (optional), and **advice expressions** (optional). The **target** defines the set of requests to which the **rule** is intended to apply in the form of a logical expression on **attributes** in the request. The **effect** of the **rule** indicates the **rule**-writer's intended consequence of a "True" evaluation for the **rule**. Two values are allowed: "Permit" and "Deny". **Condition** represents a boolean expression that refines the applicability of the **rule** beyond the **predicates** implied by its **target**. When a **PDP** evaluates a **rule** containing **obligation** expressions, it evaluates the **obligation** expressions into **obligations** and returns certain of those **obligations** to the **PEP** in the response **context**. When a **PDP** evaluates a **rule** containing **advice** expressions, it evaluates the **advice** expressions into **advice** and returns certain of those **advice** to the **PEP** in the response context.

- **Policy**: a **PAP** combines **rules** in a **policy**. A **policy** comprises five main components: a **target**, a **rule-combining algorithm-identifier**, a set of **rules**, **obligation** expressions and **advice** expressions. The **rule-combining algorithm** specifies the procedure by which the results of evaluating the component **rules** are combined when evaluating the **policy**.

- **Policy set**: comprises five main components: a **target**, a **policy-combining algorithm-identifier**, a set of **policies**, **obligation** expressions, and

advice expressions. The **policy-combining algorithm** specifies the procedure by which the results of evaluating the component **policies** are combined when evaluating the **policy set**, i.e. the `Decision` value placed in the response **context** by the **PDP** is the result of evaluating the **policy set**, as defined by the **policy-combining algorithm**. A **policy set** may have combining parameters that affect the operation of the **policy-combining algorithm**.

The XACML syntax is defined in a schema associated with the following XML namespace:

`urn:oasis:names:tc:xacml:3.0:core:schema:wd-17`

Figure 14 shows an XACML **policy** example. The policy states "Any user with an e-mail name in the "med.example.com" namespace is allowed to perform any **action** on any resource".

Figure 14. XACML policy example

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy
  xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
  http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
  PolicyId="urn:oasis:names:tc:xacml:3.0:example:SimplePolicy1"
  Version="1.0"
  RuleCombiningAlgId="identifier:rule-combining-algorithm:deny-overrides">
  <Description>
    Medi Corp access control policy
  </Description>
  <Target/>
  <Rule
    RuleId="urn:oasis:names:tc:xacml:3.0:example:SimpleRule1"
    Effect="Permit">
    <Description>
      Any subject with an e-mail name in the med.example.com domain can perform any
      action on any resource.
    </Description>
    <Target>
      <AnyOf>
        <AllOf>
          <Match
            MatchId="urn:oasis:names:tc:xacml:1.0:function: rfc822Name-match">
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string"
```

```
            >med.example.com</AttributeValue>
          <AttributeDesignator
            MustBePresent="false"
            Category="urn:oasis:names:tc:xacml:1.0:subject-category: access-subject"
            AttributeId="urn:oasis:names:tc:xacml:1.0:subject: subject-id"
            DataType="urn:oasis:names:tc:xacml:1.0:data-type: rfc822Name"/>
          </Match>
        </AllOf>
      </AnyOf>
    </Target>
  </Rule>
</Policy>
```

Also, Figure 15 and Figure 16 show a ***request context*** example and the corresponding ***response context*** example, respectively. The ***access*** request that generates the ***decision request*** may be stated as follows: "Bart Simpson, with e-mail name 'bs@simpsons.com', wants to read his medical record at Medi Corp." As a result of evaluating the ***policy***, there is no ***rule*** in this ***policy*** that returns a "Permit" result for this request. The ***rule-combining algorithm*** for the ***policy*** specifies that, in this case, a result of "NotApplicable" should be returned.

Figure 15. Request context example

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
>http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
  ReturnPolicyIdList="false">
  <Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:
 >access-subject">
    <Attribute IncludeInResult="false"
      AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id">
      <AttributeValue
        DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"
         >bs@simpsons.com</AttributeValue>
    </Attribute>
  </Attributes>
  <Attributes
    Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
    <Attribute IncludeInResult="false"
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
         >file://example/med/record/patient/BartSimpson</AttributeValue>
    </Attribute>
```

```
    </Attributes>
    <Attributes
      Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
      <Attribute IncludeInResult="false"
         AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
           >read</AttributeValue>
      </Attribute>
    </Attributes>
  </Request>
```

Figure 16. Response context example

```
<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
>http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd">
  <Result>
    <Decision>NotApplicable</Decision>
  </Result>
</Response>
```

See also

For other aspects of XACML, such as extensibility points, data types and functions, combining algorithms, etc. See http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.html.

4.2. SQL Injections

One of the main security issues that have affected and continue to affect DBMS when we are implementing web applications are attacks using SQL injections (SQLI).

An SQL injection is a method of attack that allows unauthorised access to a database through a software vulnerability in the validation of data entries from the web application that exploits the database.

These are malicious – as opposed to accidental – attacks, given that the attacker is fully aware of what he or she is doing. An SQL injection attack (SQLIA) occurs when an attacker changes the desired effect of an SQL query by inserting new SQL keywords or operators into the query.

So many web applications have been affected that a great deal of attention has focused on raising awareness of the reasons for these attacks and on recommendations to prevent them from happening.

Interesting Resources

Oracle Tutorial, "Defending Against SQL Injection Attacks!":

http://download.oracle.com/oll/tutorials/SQLInjection/index.htm

SQL injection, from Microsoft:

http://msdn.microsoft.com/en-us/library/ms161953.aspx

IBM, Web application security: Testing for vulnerabilities:

http://www.ibm.com/developerworks/web/library/wa-appsecurity/#resources

Imperva, SQL injection:

http://www.imperva.com/resources/glossary/sql_injection.html

How to stop SQL injection:

http://blog.imperva.com/2012/01/sql-injection.html

To understand how SQLI works, you will need to remember that web applications have a three-tier architecture in which the browser acts as a client, the web server as an application server and the DBMS, obviously, carries out the database server functions.

There are different mechanisms of injection to produce an attack: injection through user data input, injection through cookies, injection through server variables and second-order injections (the attacker enters a malicious input that will allow him or her to indirectly activate a SQLIA when used later).

Example of an injection through data input

A user wants to obtain the bank account number of a customer by issuing a query. To do so, that user must provide both the customer's identifier (ID) and password (pwd), since this information is confidential and can only be obtained after authentication. As we are in a web application, the query is dynamically generated on the web server only once the parameters are provided by the user, and before it can be sent to the database server. The usual pattern for generating such a query is:

```
Query = "SELECT bank_account_num FROM customer WHERE id='" + identi-
fier + "'AND pwd='" + password + "';"
```

Let's suppose that the user enters "Carles" as his identifier and "wrndi9la" as his password. The dynamically generated SQL statement will be:

```
SELECT bank_account_num FROM customer WHERE id='Carles' AND pwd =
'wrndi9la';
```

This will only give the user the number of his bank account.

A hacker wishing to take advantage of the lack of validation in this input data can run the query by entering "' OR 1=1 --" as the identifier and anything (since it is irrelevant) as the password. The query generated,

```
SELECT bank_account_num FROM customer WHERE id='' OR 1=1 --' AND pwd
= '';
```

obtains the bank account number of every customer, since the user now forces the `WHERE` clause to evaluate as always true `(OR 1=1)`, regardless of the values entered as the identifier and password. Note that "—" is the way in which comments are indicated in SQL, so anything entered afterwards is ignored by the DBMS.

The different attacks can also be characterised by the aim of the hacker. The most common aims detected include:

- Extracting data is the most common type of SQLIA. The purpose is to extract data values from the database (information such as credit card numbers can be obtained) that could be of great value to the hacker.

- Adding or modifying data is done to add or change data values in the database.

- When performing database fingerprinting, the hacker attempts to discover technical information about the database, such as the type and version that a specific web application is using. Certain types of DBMS respond differently to attack queries and this information can be used to identify the DBMS. Once the hacker knows the type and version of the DBMS, he or she can organise a specific attack on the database.

- With bypassing authentication, intruders attempt to bypass the authentication mechanisms in order to assume rights and privileges associated with other users of the application.

- With identifying injection parameters, a web application is explored to discover which parameters and user input fields are vulnerable to SQLIA.

- The aim of attacks that determine the database schema is to obtain all the information on the database schema (such as names of tables, names of columns and data types in columns). This information can then be used by a hacker to successfully extract data from the database.

- With evading detection, hackers try to avoid auditing and intrusion detection mechanisms.

- Denial of service is used to prevent others from using the service offered by the database, either by shutting down the database or by blocking or deleting tables.

- The aim of executing remote commands is to execute arbitrary commands on the database. These commands can either be stored procedures or functions available to the users of the database. This type of attack is the most dangerous because it can allow a hacker to take control of the entire system.

- With privilege escalation, the attack takes advantage of errors to increase the hacker's privileges.

Vulnerability scanner

An automated tool called a vulnerability scanner can be used to pinpoint some of the potential vulnerabilities of a web application.

The attacks used to achieve the various aims of the hackers, whether in isolation or in conjunction with other attacks, are classified as:

1) Tautologies. The goal of these attacks is to inject code into the SQL statements so that the set of conditions of the `WHERE` clause will always evaluate to true (for example, "`OR 1=1`"). This allows the hacker to bypass identification and extract data as well as identifying injectable parameters.

2) Illegal/Logically incorrect queries. The hacker generates incorrect queries in order to obtain information from the error messages returned by the system, since they contain useful information. These error messages help the hacker to find the vulnerable parameters of the application and determine the database being used.

3) Union query. With this technique, hackers inject a new query into the secure query using the `UNION` statement, which brings up different data to that accessed by the original query.

4) Piggy-backed queries. In this type of attack, intruders add lots of additional queries to the query that the system should have received. They do not change the original query, but instead inject a ";" followed by the rest of the queries to be added. The first query is usually legitimate but the subsequent ones are illegitimate. By adding these new queries, the hacker can obtain additional information or delete/destroy part or all of the database.

5) Stored procedures. The stored procedure adds additional functionality to the database. Once hackers are able to identify the database behind a web application, they can use injection in stored procedures to continue the attack and obtain greater privileges, execute remote commands and even affect the availability of the services.

6) Alternate encodings. With this technique, the hacker modifies the query by encoding it (e.g. in hexadecimal, ASCII or Unicode) so that it goes unnoticed by automated prevention techniques (which are based on the recognition of special characters detected previously in SQL injections).

7) Inference. Intruders use this attack when the application is sufficiently secure and error messages cannot be obtained. The purpose of the changes made to queries is to obtain true/false responses. Depending on the responses to the different queries, certain information can be deduced. There are two techniques for this attack:

- Blind injection: true/false queries are used and depending on the response, the system's operation continues as normal or is altered.
- Timing attacks: these allow a hacker to obtain information from a database by observing the delays in the database response. This technique uses an `if-then` statement for query injections, expressly including the `WAIT-FOR` builder, which causes the database to delay its response for a specific length of time depending on the options.

Bibliography

Tajpour, A.; Ibrahim, S.; Masrom, M. (2011). *SQL Injection Detection and Prevention Techniques*. International Journal of Advancements in Computing Technology (Volume 3, Number 7, August).

Different techniques have been proposed to help combat SQL injections. For example, a series of defensive coding best practices have been established, although their success always depends on the absence of any human error. Techniques have also been devised for both detection and prevention, although it is difficult that any one of the techniques in isolation can deal successfully with every type of attack.

5. Data Protection Act

Besides the importance of guaranteeing data security for the sake of our businesses, we must also comply with the existing legal framework for the protection of personal data.

Spanish current Organic Law 15/1999, dated December 13th 1999, on the Protection of Personal Data (Data Protection Act), transposed into Spanish law Directive 95/46/EC of the European Parliament and of the Council of October 24th 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, repealing Organic Law 5/1992, of October 29th 1992, governing the processing of personal data.

Royal Decree 1720/2007, of December 21st 2007, subsequently approved the implementing regulations of Organic Law 15/1999, of December 13th 1999, on the protection of personal data and repealed Royal Decree 1332/1994, of June 20th 1994, implementing certain aspects of Organic Law 5/1992, of October 29th 1992, governing the automated processing of personal data, Royal Decree 994/1999, of June 11th 1999, approving the regulation on security measures for automated files containing personal data, and all regulations of equal or lower rank contrary to or contradicting the provisions of this Royal Decree.

These regulations share the aim of the Data Protection Act in addressing the potential risks for person rights of the collection and processing of personal data.

The Act, after specifying its scope and defining the basic concepts, describes the principles of data protection: data quality, the right to be informed in data collection, consent of the concerned party, specially protected data, data on health, data security, duty of confidentiality, data communication and data access by third parties. It then moves on to deal with the rights of individuals: contesting of assessments, right to consult the General Data Protection Register, right of access, right of rectification and cancellation, procedure of opposition, access, rectification or cancellation, protection of rights, and entitlement to compensation.

This section is followed by the sectoral rules regarding public and privately-owned files and makes reference to the general rule and to exceptions to the international data circulation. The Act subsequently introduces the Spanish Data Protection Agency and the offences and penalties, dividing these offences into minor, serious and very serious.

The **minor offences** are:

a) Not sending to the Spanish Data Protection Agency the notifications under the Act or its implementing provisions.

b) Not applying to enter the personal data file in the General Data Protection Register.

c) Violation of the duty to inform the party concerned about the processing of their personal data when the data are requested by the interested party.

d) Transmitting personal data to a processor without fulfilling the formal duties set out in Article 12 of the Act.

The **serious offences** are:

a) Creating public files or beginning to collect personal data for these without authorisation from a general provision published in the Official State Gazette (BOE in the case of Spain) or relevant official journal.

b) Processing personal data without asking for the consent of the individuals concerned when this is necessary to comply with the provisions of the Act and its implementing regulations.

c) Processing personal data or using them subsequently with infringement of the principles and guarantees set out in Article 4 of the Act and its implementing regulations, except when these constitute a very serious offence.

d) Violation of the duty of secrecy in relation to personal data processing, referred to in Article 10 of the Act.

e) Obstructing or preventing exercise of the rights of access, rectification, cancellation or opposition.

f) Violation of the duty to inform the party concerned of the processing of their personal data when the data have not been obtained from the interested party.

g) Violation of other obligations or requirements to notify the party concerned, as imposed by the Act and its implementing regulations.

h) Keeping files, premises, programs or equipment containing personal data without the due security conditions, as determined in regulatory proceedings.

i) Failing to meet the requirements or heed the warnings of the Spanish Data Protection Agency or failing to supply all of the documents and information requested by the latter.

j) Obstructing inspections.

k) Notifying or transferring personal data without being entitled to do so under the terms provided in this Act and its implementing regulations, except where this constitutes a very serious offence.

The **very serious offences** are:

a) The misleading or fraudulent collection of data.

b) Processing or transferring the personal data referred to in paragraphs 2, 3 and 5 of Article 7 of the Act, except where authorised by the Act, or infringing the prohibition contained in paragraph 4 of Article 7.

c) Failing to stop the illegal processing of personal data in the event of prior notification by the Director of the Spanish Data Protection Agency to this effect.

d) The international transfer of personal data to countries that do not provide a comparable level of protection without authorisation from the Director of the Spanish Data Protection Agency except where, in accordance with the Act and its implementing regulations, such authorisation is not required.

The Act introduces a wide range of penalties for various offences. For example, minor offences are punished with a fine of EUR 900 to 40,000, serious offences are fined EUR 40,001 to 300,000 and very serious offences are fined EUR 300,001 to 600,000.

The implementing regulations follow the structure of the contents of the Act and place special emphasis on security measures for personal data processing, classifying them into three levels (basic, medium and high), and on the security document that must include all of the information about existing files and the adopted measures. The regulations end by listing all procedures handled by the Spanish Data Protection Agency.

All files and acts of personal data processing must adopt the security measures classed as basic-level. In addition, medium-level security measures must be applied to the following files or acts of personal data processing:

a) Those related to administrative or criminal offences.

b) Those whose operation is governed by Article 29 of Organic Law 15/1999 of 13 December 1999.

c) Those for which tax authorities are responsible and which are related to the exercise of their tax powers.

d) Those for which financial institutions are responsible for purposes related to the provision of financial services.

e) Those for which management entities and common social security services are responsible and which are related to the exercise of their powers. Likewise, those for which for mutual societies for occupational accident and diseases of the social security service are responsible.

f) Those containing a series of personal data that give a definition of the characteristics or personality of citizens and can be used to evaluate certain aspects of their personality or behaviour.

Besides the basic and medium-level measures, high-level measures must also be applied to these:

a) Those referring to data on ideology, trade-union membership, religion, beliefs, race, health or sexual orientation.

b) Those containing or referring to data obtained for law enforcement purposes without the consent of the individuals concerned.

c) Those containing data derived from acts of gender violence.

Nonetheless, the regulation sets down certain special circumstances in which the application of security measures can be relaxed.

Summary

In this module, we have learnt that systems that use databases are vulnerable and can be subject to various threats. In order to make databases secure, we have learnt about the various security controls that can be adopted and the possible mechanisms for intrusion detection.

With regard to making decisions on how to protect information, we have discussed the main security models: discretionary access control, mandatory access control and role-based access control. We also introduced statistical databases, as a specific case of the need for inference control.

Given recent technological advances, we looked at two specific cases requiring special attention: access control for XML and SQL injection, the latter being a major threat to web systems today.

Lastly, we highlighted the most important points of the Data Protection Act in order to make students aware of the need to properly protect personal data that can be stored on digital media.

Activities

1. Explain which security models are supported by Oracle.

Self-evaluation

1. What is the main difference between discretionary access control and role-based access control?

2. Assuming the same database as used in the example in section 2.1 and considering that the DBMS uses a discretionary access control system:

```
library(code, address, city, telephone, capacity)
book(ISBN, title, publisher, publication_year)
person(ID, name, title, address, city, birth_year)
author(ISBN, ID)
      FOREIGN KEY (ISBN) REFERENCES book(ISBN),
      FOREIGN KEY (ID) REFERENCES person(ID),
has(code, ISBN, quantity)
      FOREIGN KEY (code) REFERENCES library(code),
      FOREIGN KEY (ISBN) REFERENCES book(ISBN),

GRANT INSERT, DELETE, UPDATE ON library, book, has TO library_admin;
GRANT ALL PRIVILEGES ON has TO Joan;
GRANT ALL PRIVILEGES ON library TO library_super_admin WITH GRANT OPTION;
REVOKE SELECT ON has TO Joan;
```

Write the SQL statements necessary to:

a) Grant the right to grant query privileges on the *book* table to a user identified as *Joan*.

b) Revoke all privileges granted to the user *Pere*.

c) Grant privileges to insert and delete tuples in the *author* and *has* tables to the users *Maria, Carme* and *Pau*.

d) Could any user of the database execute the statements you wrote in the above sections?

3. In a DBMS with mandatory access control, taking into account the *Employee* multilevel relation in Figure 4, how would the data of the relation be affected if a TS-level user executed the statement:

```
UPDATE Employee
SET Dept= 'Dept2', Sou = '20.000'
WHERE nom = 'Bernat';
```

4. With the same DB as in Exercise 2, assuming that the DBMS performs role-based access control, using SQL statements:

a) Establish a hierarchy of roles and necessary authorisations for each role so that the roles defined are able to perform the following tasks:

- *Librarian* role: insert and delete tuples in the *book*, *author* and *has* relations.
- *Manager* role: insert, delete and update tuples in the *library*, *book*, *person*, *author* and *has* relations.
- *Head* role: insert, delete and update tuples in the *book*, *author* and *has* relations.

b) Assign the *Manager* role to the user *Paula* and remove the user *Miquel* from the *Head* role.

Answer key

Self-evaluation

1. In DAC, authorisations are granted to different users of the DB, while in RBAC the authorisations are granted to roles. These roles are defined in terms of a set of tasks specific to the role. Users are assigned to a role and hence, obtain authorisations through this.

2. a) `GRANT SELECT ON book TO Joan WITH GRANT OPTION;`

b) `REVOKE ALL PRIVILEGES FROM Pere CASCADE;`

c) `GRANT INSERT, DELETE ON author, has TO Maria, Carme, Pau;`

d) Only the creator (owner) of each relation (table), or a user who had previously received privileges and had not had them revoked at the time of execution could execute the statements.

3.

Instance at level S of multilevel Employee relation

Name	C_{Name}	Dept.	C_{Dept}	Salary	C_{Salary}	TC
Bernat	S	Dept1	S	10.000	S	S
Anna	S	Dept2	S	–	S	S

Instance at level TS of multilevel Employee relation

Name	C_{Name}	Dept.	C_{Dept}	Salary	C_{Salary}	TC
Bernat	S	Dept1	S	10.000	S	S
Bernat	S	Dept2	TS	20.000	TS	TS
Anna	S	Dept2	S	20.000	TS	TS
Sara	TS	Dept2	TS	30.000	TS	TS

4. a)

```
CREATE ROLE librarian;
GRANT INSERT, DELETE ON book, author, has TO librarian;
CREATE ROLE head;
GRANT librarian TO head;
GRANT UPDATE ON book, author, has TO head;
CREATE ROLE manager;
GRANT head TO manager;
GRANT INSERT, DELETE, UPDATE ON library, person TO manager;
```

b)

```
GRANT manager TO Paula;
REVOKE head FROM Miquel;
```

Glossary

Access control deals with preventing unauthorised operations on the managed data.

Authentication mechanisms of ensuring that entities are who they claim to be.

Authorisation the function of specifying access rights to data.

Database auditing retains a secure record of database operations that can be used to verify compliance with desired security policies, to trace policy violations, or to detect anomalous patterns of access.

Database security a discipline that seeks to protect data stored in a DBMS from intrusions, improper modifications, theft and unauthorised disclosures.

Discretionary Access Control (DAC) provides for owner-controlled administration of access rights to objects. A DAC mechanism allows users to grant or revoke access to any of the objects under their control.

Inference control a discipline that seeks to protect data so they can be published without revealing confidential information that can be linked to specific individuals among those to which the data correspond. Also known as **Statistical Disclosure Control (SDC)**.

Intrusion Detection (ID) the process of monitoring events occurring in a system and signalling responsible parties when interesting (suspicious) activity occurs.

Mandatory Access Control (MAC) a kind of access control defined by the National Computer Security Center's Trusted Computer System Evaluation Criteria (TCSEC) as 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 authorisation (i.e. clearance) of subjects to access information of such sensitivity. Also known as **Multilevel Security**.

Role Based Access Control (RBAC) involves controlling access to computer resources and information by defining users, roles, and permissions, and assigning users and permissions to roles. Also known as **Role Based Security**.

Rounding (or Microdata rounding) a family of masking methods for statistical disclosure control of numerical microdata; a similar principle can be used to protect tabular data. It replaces original values of attributes with rounded values.

Security mechanisms provide the appropriate desired security functionality according to the security criteria established for the database.

SQL-Injection Attacks (SQLIAs) these attacks may allow the intruder to gain access to all the data stored into the database by bypassing the security mechanisms. They therefore give him or her the power to leak, modify or delete the information stored in the database.

Statistical Database a database designed to explicitly handle "macro data," i.e. data computed by different forms of summarisation, including grouping and classification, as first-class objects. One of the goals is to hide the micro data (i.e. the raw data records from which the macro data are computed) from user queries.

Bibliography

Autoritat Catalana de Protecció de Dades (Catalan Data Protection Agency) http://www.apd.cat/ca/index.php (last visited March 2012)

Bertino, E.; Sandhu, R. (2005). *Database Security – Concepts, Approaches, and Challenges*. IEEE Transactions on Dependable and Secure Computing. Vol 2, N. 1, January-March.

Castano, S.; Fugini, M.; Martella, G.; Samarati, P. (1995). *Database Security*. Addison-Wesley.

Organic Law 15/1999, of 13 December 1999, on the protection of personal data (BOE – Official State Gazette – No. 298, of 14/12/1999).

Liu, L.; Özsu, M.T. (Eds.) (2009). *Encyclopedia of Database Systems*. Springer.

Halfond, W.G.J.; Viegas, J.; Orso, A. (2006). *A Classification of SQL Injection Attacks and Countermeasures*. International Symposium on Secure Software Engineering (ISSSE 2006).

Patel, N.; Mohammed, F.; Soni, S. (2011). *SQL Injection Attacks: Techniques and Protection Mechanisms*. International Journal on Computer Science and Engineering (IJCSE). Vol. 3 No. 1, pp. 199-203.

Royal Decree 1720/2007, of 21 December 2007, on the protection of personal data (BOE – Official State Gazette – No. 17, of 19/01/2008).