Medical Informatics Europe '97
C. Pappas et al. (Eds.)
IOS Press, 1997

Implementing Security On a Prototype Hospital Database

Marie KHAIR (1), George PANGALOS (2), Foteini ANDRIA (2), Lefteris BOZIOS (3)
(1) Computers Department, Faculty of Sciences, Notre Dame University, Zouk Mosbeh, PO Box 72, Zouk Mikael, Lebanon. mkhair@ndu.edu.lb
(2) Computers Division, Faculty of Technology, General Department, Aristotle University of Thessaloniki, 540 06 Thessaloniki, Greece. gip@eng.auth.gr, fanr@egnatia.ee.auth.gr
(3) Information Technology Dept., AHEPA Hospital, 546 36 Thessaloniki, Greece

Abstract. This paper describes the methodology used and the experience gained from the application of a new secure database design approach and database security policy in a real life hospital environment. The applicability of the proposed database security policy in a major Greek general hospital is demonstrated. Moreover, the security and quality assurance of the developed prototype secure database is examined, taking into consideration the results from the study of the user acceptance.

1. Introduction

A hospital is a typical case of a security critical environment [3], [4]. Due to the widespread use of the database technology and its role and nature, database security plays today a significant role in the overall security of health care information systems [5]. The development of a secure database system requires an appropriate multiphase design methodology which will guide the steps of the development and will provide tools supporting the automatic execution of some steps [1]. Such a design methodology has been proposed in [4] and [5].

In this paper we will focus on the problems related to the application of this methodology and the experience gained from its experiment implementation in a major Greek general hospital.

2. Overview of the proposed secure database design methodology

The two most well known database security policies are the mandatory and the discretionary one [1], which both have proved to be insufficient to cover the security needs of the health care environments [5]. The proposed methodology and security policy is based on the integration of mandatory and discretionary security policies [3],[4],[5]. A step by step design methodology with integrated security has been proposed. The responsibility of the role in the application determines the security label (clearance) of the user role. The user roles are assigned a node at the user role hierarchy. According to the data the user roles need to access, security labels (classification) are assigned to data. Polyinstantiation is supported only in the form of cover stories, due to the support of the write down mechanism (with no fear of inference), which is essential for the hospital environment. A detailed description of the proposed methodology can be found in [4], [5].

3. Pilot Implementation of the Proposed Security Policy

The AHEPA general university hospital was used as a testbed for the implementation of our proposed secure database design methodology in a real life hospital environment. There is a long-standing cooperation between the informatics laboratory and the EDP unit of the AHEPA hospital [2], [4], [5]. Following, we will describe briefly the information system of the hospital and the functions which the prototype secure hospital database SEC_AXEPA was designed to serve.

3.1. Implementation of the prototype secure hospital database SEC_AXEPA

The integrated information system of AHEPA hospital covers the patient administration system (inpatient-outpatient), billing, financial, personal management and payroll. It has been running since 1990 and operates on an inter network of distributed databases based on the client-server model. The application SEC_AXEPA has been developed in order to serve the in-patient administration purposes. This implementation will be used as a pilot for the subsequent implementation of the whole secure hospital information system.

The prototype hospital database SEC_AXEPA was implemented on Ingres, a relational DBMS (since there was not a Trusted DBMS available). The data sets loaded on the prototype hospital database are: the nurse record, the doctor record, the personal patient information record , the patient medical record, the laboratory information and the follow-up information. During his/ her hospitalisation, the patient identification is based on a serial number assigned to him/her, when he/she checks in.

Being a relational database management system, Ingres stores data in tables. The labelling of the data sets can change dynamically, if one of the security constraints is satisfied, leading to upgrading and/or fragmentation. The classification level of each tuple of a table was implemented by adding at each table a column, which was only visible to the D.B.A., and contained the tuple classification. That is 1,2,3 and 0 respectively for confidential, secret, top secret and cover story. Another additional column was added to the related medical data tables, containing the flag 'h' (history) or 'l' (last). It is automatically updated, when new data was inserted. This is one of the integrity constraints that were implemented in the prototype database.

When a user accesses the database, a corresponding user interface is automatically presented to him/her, based on the user group (doctor or nurse) where he/she belongs to. We also support two user roles for the nurse group; the special nurse and the normal nurse, which are given respectively a high and a low level of clearance. The use of Ingres rules and procedures, that create convenient cover stories, had made it possible for these user roles to have the same user interfaces in the application. The security labels which were defined during the secure conceptual design phase were implemented using the notions of roles and groups supported by Ingres. After performing a study on the considered users roles (doctor, nurse, and normal nurse), clearance level 3, 2 and 1 were respectively assigned them.

The access types supported from the secure prototype application are: insert, read, update, execute, cancel. Thus, the possibility of deleting information is excluded, not only for reasons of better control of the information flow, but also for reducing the possibility of fatal mistakes. For the better understanding of the implementation, we will describe next the processing of the security constraints.

3.2. Security Constraints Processing

3.2.1. Definition of the security constraints

The security classification constraints are specified by the database designer in co-operation with the application users [6]. Their application and processing may result to fragmentation and/or upgrading of the classification of some data sets in the medical database [5]. It must be noted that we will use the illness HIV (AIDS) as a representative of every sensitive illness and the characterism VIP (Very Important Person) for the people whose medical record is considered sensitive (that is important people, hospital staff personnel, political, and other well known persons, etc.). Moreover we will represent the unclassified level by 1, the secret by 2, and the top secret by 3.

Below we will define the security constraints that have been applied to the secure prototype application SEC_AXEPA. First we present them informally (as they were defined by the users), we categorise them and then we transform them into mathematical language.

♦ *Constraint 1*: The sensitivity level of the name determines the sensitivity level of all the patient personal information. Level-Based Constraint, LbC(PAT-PERS-INFO,{*},name).

♦ *Constraint 2:* The name of the patient is always considered top secret, so the personal information is always considered top secret. Simple Constraint, SiC(PAT-PERS-INFO,{name},3). Then, SiC(PAT-PERS-INFO,{*},3). stands for both constraints 1 and 2.

♦ *Constraint 3*: The data concerning the disease in the patient medical record is considered secret, if the disease is HIV.

Content-Based Constraint, CbC(Diagnoses,{*},Diagnosis,'=','HIV',2).

♦ *Constraint 4*: If the lab result is HIV, then the information concerning the lab-test is considered secret. Complex Constraint, CoC(PAT-LAB-EXAMS,{*},LAB-EXAM-RESULT,'=','HIV',2).

♦ *Constraint* 5: If the lab result is HIV, then the information concerning the diagnosis is considered secret. Complex Constraint, CoC(Diagnoses,{Diagnosis),LAB-TEST-RESULT,Lab-exam-result,'=','HIV',2).

♦ *Constraint 6*: A nurse should always be able to look at the patient's diagnosis (no matter what his/her status is). This is supported, in order he/she to be able to offer convenient care to the patient and at the same time take himself/herself suitable precautions. In the case of a sensitive diagnosis (see constraint 5), a suitable cover story is created.

♦ *Constraint 7*: If the patient is a VIP, then no matter the diagnosis for the patient, it is considered secret and a suitable cover story is created, in order to prevent normal nurses to infer sensitive information. Complex Constraint, CbC(Diagnoses,{Diagnosis}, PAT-PERS-INFO,P-status,'=','1',2) AND cover story.

♦ *Constraint 8:* The property ICD9-code of diagnosis is considered secret, while the values of ICD9-code, without information as to whom they refer to, are considered unclassified.

Association-Based Constraint , AbC(Diagnoses, {ICD9-code}, 2).

♦ *Constraint 9*: The information of the prescription of a doctor to a certain patient is considered unclassified. However, the aggregation of the prescriptions given to a patient is considered secret, since it may indicate the changes in the patient's health condition.

Aggregation Constraint, AgC(Prescribes, {Doses}, '3', 2).

♦ *Constraint 10*: The laboratory exam results are considered unclassified.. In that way, a lab staff member can access all the results of the exams, even if they concern a sensitive disease

(HIV). However, if that user accesses the corresponding diagnosis made for the same patient and realise that it is different from the HIV, he/she can infer the existence of cover stories. This problem is solved by assigning to users and data apart from levels of sensitivity, disjoint categories. Inference Constraint, IfC(Diagnoses, {Diagnosis}, LAB-TEST-RESULT, {lab-exam-result}, corresponding category}.

Since we support tuple-level granularity, level-based constraints automatically undertake the entity integrity constraints. We must also note that apart from the security constraints, integrity constraints and availability constraints were also considered during the implementation. However, since we believe that they are rather typical of a hospital database, we have chosen not to refer to them in this paper.

3.2.2. Implementation of the Security Constraints using SQL

For the implementation of the security constraints that were defined earlier for the hospital database, the security features, rules and procedures supported by Ingres were used. We have applied in our experimental implementation all the different kinds of security constraints to the prototype hospital database SEC_AXEPA. We will describe now a typical implementation of each one type of such a security constraint. We note that since we support tuple-level granularity, the level-based constraints and the entity integrity constraints are automatically satisfied.

A *simple security constraint (*SiC) is implemented by creating a rule that is fired each time new data is inserted. This rule executes a procedure that sets the tuple class of the data inserted equal to the desired classification level. The implementation of the *content-based constraints* is similar. The following example also includes the implementation of the cover story created.

Example implementation of a content-based security constraint and of a Cover Story

SCL: CbC(Diagnoses,{*},Diagnosis,'=','HIV',2) AND cover story

Rule 3: Create rule r_3 after insert of diagnoses execute procedure p_3
(pid = new.p_id, doctor = new.dr_code, diagdate= new.diag_date, diagnosis = new.diag);

Procedure 3: Create procedure p_3 (pid integer4, drcode c(15), diagdate date, diagnosis c(500)) as
begin if diagnosis = 'HIV' then update 'marie'.diagnoses set tc = 2 where p_id = :pid and
diag_date = :diagdate and dr_code = :drcode and diag = :diagnosis;
insert into 'marie'.diagnoses(p_id, dr_code, diag_date, diag, icd9_code, diag_comm, fl, tc)
values (:pid, :drcode, :diagdate, 'Blood disease', '222', 'Take care', 'l', 0); endif; end;

The *complex-constraint* implementation is executed in rather the same way.

Example implementation of a complex security constraint

SCL: CoC(PAT-LAB-EXAMS,{*},LAB-EXAM-RESULT,'=','HIV',2)

Rule 4: Create rule r_4 after insert of lab_test_res execute procedure p_4
(pid = new.p_id, labexams = new.lab_exam_res, resdate = new.res_date);

Procedure 4: Create procedure p_4 (pid integer 4, labexamres c(100)) as begin
if labexamres = 'HIV' then update 'marie'.lab_test_res set tc = 2 where p_id = :pid; endif; end;

An *association-based security constraint* implemented on Ingres is equivalent to an aggregation constraint, since column-level control is supported only in the update access mode in Ingres. An *aggregation constraint* is implemented by setting a limit to the data the user is able to retrieve. Finally, the *inference constraint's* implementation is carried out by assigning different categories to the data, so that a user can not access certain data and infer sensitive information.

4. Study of the User Acceptance

As part of the experimental implementation, we also studied the user acceptance of the system. The first general conclusion is that most hospital users agreed that security is very important at a hospital. They would also prefer to use a more complex and secure information system, provided that they would be trained properly. They defined, however, the notion of database security in several ways. Moreover, they believed that confidentiality and integrity contribute more to the security of a hospital database than availability. As far as the specific database SEC_AXEPA is concerned, most doctors and nurses thought that using it would significantly improve their work. Additionally, they found it easy to use and satisfyingly secure. The support of cover stories and the user role hierarchy were found rather complex, but important and necessary for a hospital database. They all agreed however that it is important to define legally soon the various types of data sets and user roles in a hospital.

To conclude, almost all the users of the SEC_AXEPA hospital database were positive towards the support of the proposed security environment and the prototype secure hospital database itself.

5. Conclusions

The implementation of the prototype hospital database has shown that the proposed secure database design methodology can be successfully applied at a complex real-life hospital environment. It has also been shown that in the prototype hospital database all the three aspects of security were ensured with no significant overheads. More specifically, the secrecy (confidentiality) was preserved by allowing only the doctors to access the patient's personal data and by creating cover stories for important people. Integrity was also preserved by providing for the users with the least possible privileges and by using security levels. Finally, availability was also well preserved; the performance overheads were relatively low, priority was assigned to the user transactions depending on the user's role in the hospital and flags were used to allow the users to have direct access to the most recent medical data .

We believe therefore that the lessons learned from the implementation of the prototype secure hospital database SEC_AXEPA will be useful to other database designers, who will attempt a similar task in the future.

6. References

[1] S. Castano, M. Fugini, G. Martella, P. Samarati, Database security, Addison Wesley publishing company, 1994.

[2] G. Pangalos, M. Khair, L. Bozios, Enhancing medical database security, In: Journal of medical systems, USA, 1994.

[3] G. Pangalos S. Katsikas, and D. Gritzalis, Medical database security guidelines, In: Computers and security journal, 1994.

[4] G. Pangalos, M. Khair, L. Bozios, An integrated secure design of a medical database system, In: MEDINFO'95, The 8th world congress on medical informatics, Canada, 1995.

[5] G. Pangalos, M. Khair, Design of secure medical database systems, In: IFIP/SEC'96, 12th international information security conference, 1996.

[6] G. Pernul, Database security, Academic Press Inc., 1994.