Secure Database Development

Jan Jurjens (1) and Eduardo B. Fernandez (2)

(1) Computing Department, The Open University, Milton Keynes, MK7 8LA GB http://www.jurjens.de/jan
(2) Dept. of Computer Science, Florida Atlantic University, Boca Raton, FL 33431, USA
http://www.cse.fau.edu/~ed

Synonyms
Secure DBMS development, secure database design

DEFINITION
This entry considers how to build secure database system software. In particular, it describes how to build a general-purpose database management system where security is an important design parameter. For the database community the words secure database design may refer to the schema design to produce a database for a specific application with some level of security properties. There is a large amount of literature on this latter subject and a related section in this encyclopedia (Database security). This section concentrates mostly on how to build the software of a DBMS such that it exhibits security properties, which is called secure database development. Both approaches are contrasted so that the reader can decide which one of these problems applies to her specific case but more space is dedicated to the general secure database development problem.

HISTORICAL BACKGROUND
While there is a large number of papers on security models including authorization and other security aspects of databases [2, 4, 6], there is little work on how to implement a secure Database Management System (DBMS). It is true that many proposals for secure multilevel databases include details of implementation but most of them are ad hoc architectures that cannot be generalized to databases using different models or even to other multilevel databases with different requirements. Of the books on database security, [5] had several chapters on how to build secure relational database systems, and later [4] included also multilevel models. Those books do a good job of indicating the architectural units of such systems and their general requirements. However, software development aspects are not discussed in detail. It appears that [10] is the only work discussing these aspects explicitly.

SCIENTIFIC FUNDAMENTALS
There are two aspects to the problem of developing software for secure databases: building a general (application-independent) secure DBMS and building a database system which is part of a secure application. These two problems are first briefly defined and then discussed in more detail. Other approaches and possible system architectures are also considered.

In the first approach the DBMS is just a complex software application in itself and a general secure software methodology can be applied without or with little change. Object-oriented applications typically start from a set of use cases, which define the user interactions with the system under development. In this particular case, use cases would define the typical functions of a DBMS, e.g. search, query, and update, and security would be included as part of its development life cycle. The DBMS would follow an appropriate model, e.g. Role-Based Access Control (RBAC), selectable in the design stage, which defines security constraints for the

functions defined by the use cases. In some cases, it may be possible to support more than one security model (the section on Authorization has details on models). This would result in a secure DBMS where security would be a general nonfunctional requirement. The approach results in a general-purpose secure DBMS, where nothing is known about the specific applications that will be executed by its users. The DBMS itself is the application. The secure development methodologies of [7,13] and others are applicable here.

Another view is the one from a designer who needs to build a specific user application (or type of application) that includes a DBMS as part of its architecture, e.g. a financial system (most applications require a database but the degree of security needed may vary). This is discussed in [4, 9, 10, 11]. In this case, the DBMS is rather ad hoc and tailored to the level of security desired for the specific type of application. For example, [11] separates the requirements into three types: functional, security, and database. Typically, these approaches emphasize how to define and enforce a set of application-specific rules that follow some security model and how to reflect them in the schema and other parts of the DBMS. Most of these studies emphasize the security of the database schema or some specific sections without much concern for the rest of the application. A methodology such as [7] or [13] can also be applied here, the DBMS being one of the architectural levels of a system that implements a specific application, although these methodologies have little to say about the contents of the specific rules that are needed in the schema (only their safe storage but not their consistency or security).

An interesting problem that applies to both approaches is the mapping from the conceptual security model (that may apply to a collection of DBMSs) to the authorization system of a specific database; for example, security constraints defined in a conceptual UML model defining authorizations in terms of classes must be mapped to an SQL-based authorization system which defines authorizations in terms of relations. Clearly, whatever is defined in the common conceptual model must be respected in the DBMS authorization system, although this latter may add further constraints related to implementation aspects.

General secure database systems

In this case, as indicated earlier, the DBMS is a complex application requiring a general high level of security. There are several methodologies for this purpose and two of them are described below. A methodology for secure software development should include appropriate tools and provide a unified and consistent approach through all the life cycle stages. Ideally, a methodology should use a Model-Driven Development approach, where transformations between development stages are based on corresponding metamodels. Since the resulting software is independent of the access control model adopted, it does not provide for special requirements of the model; for example, multilevel models typically require data labeling. This means that the resulting software would be less secure than an ad hoc design (unless the multilevel model was the target in the example). Because of the generality of the resultant DBMS it may be difficult to prove formally security properties. An early approach in this direction was based on adding security functions to a general-purpose DBMS, e.g. INGRES or System/R.

Secure Database Development using Patterns

A methodology to build secure systems is presented in [7]. A main idea in the proposed methodology is that security principles should be applied at every stage of the software lifecycle and that each stage can be tested for compliance with those principles. Another basic idea is the use of patterns at each stage. A pattern is an encapsulated solution to a recurrent problem and

their use can improve the reusability and quality of software.

Domain analysis stage: A business model is defined. Legacy systems are identified and their security implications analyzed. Domain and regulatory constraints are identified and use as global policies. The suitability of the development team is assessed, possibly leading to added training. This phase may be performed only once for each new domain or team. The need for specialized database architectures should be determined at this point. The approach (general DBMS or application-oriented system) should also de defined at this stage.

Requirements stage: Use cases define the required interactions with the system. Each activity within a use case is analyzed to see which threats are possible. Activity diagrams indicate created objects and are a good way to determine which data should be protected. Since many possible threats may be identified, risk analysis helps to prune them according to their impact and probability of occurrence. Any requirements for degree of security should be expressed as part of the use cases.

Analysis stage: Analysis patterns can be used to build the conceptual model in a more reliable and efficient way. The policies defined in the requirements can now be expressed as abstract security models, e.g. access matrix. The model selected must correspond to the type of application; for example, multilevel models have not been successful for medical applications. One can build a conceptual model where repeated applications of a security model pattern realize the rights determined from use cases. In fact, analysis patterns can be built with predefined authorizations according to the roles in their use cases. Patterns for authentication, logging, and secure channels are also specified at this level. Note that the model and the security patterns should define precisely the requirements of the problem, not its software solution. UML is a good semi-formal approach for defining policies, avoiding the need for ad-hoc policy languages. The addition of OCL (Object Constraint Language) can make the approach more formal.

Design stage: When one has defined the policies needed, one can select mechanisms to stop attacks that would violate them. A specific security model, e.g. RBAC, is now implemented in terms of software units. User interfaces should correspond to use cases and may be used to enforce the authorizations defined in the analysis stage. Secure interfaces enforce authorizations when users interact with the system. Components can be secured by using authorization rules for Java or .NET components. Distribution provides another dimension where security restrictions can be applied. Deployment diagrams can define secure configurations to be used by security administrators. A multilayer architecture is needed to enforce the security constraints defined at the application level. In each level one can use patterns to represent appropriate security mechanisms. Security constraints must be mapped between levels.

The persistent aspects of the conceptual model are typically mapped into relational databases. The design of the database architecture is done according to the requirements from the uses cases for the level of security needed and the security model adopted in the analysis stage. Two basic choices for the enforcement mechanism include query modification as in INGRES and views as in System R. A tradeoff is using an existing DBMS as a Commercial Off-the-Shelf (COTS) component, although in this case security will depend on the security of that component.

Implementation stage: This stage requires reflecting in the code the security rules defined in the design stage. Because these rules are expressed as classes, associations, and constraints, they can be implemented as classes in object-oriented languages. In this stage one can also select specific

security packages or COTS, e.g., a firewall product or a cryptographic package. Some of the patterns identified earlier in the cycle can be replaced by COTS (these can be tested to see if they include a similar pattern). Performance aspects become now important and may require iterations. As indicated, a whole DBMS could be such component.

An important aspect for the complete design is assurance. Experience shows that one can verify each pattern used but this does not in general verify their combination. One can however still argue that since one has used a careful and systematic methodology with verified and tested patterns, the design should provide a good level of security. The set of patterns can be shown to be able to stop or mitigate the identified threats.

Secure Database Development using UMLsec
A general methodology for developing security-critical software which in particular can be used to develop secure DBMSs has been proposed in [13]. It makes use of an extension of the Unified Modeling Language (UML) to include security-relevant information, which is called UMLsec. The approach is supported by extensive automated tool-support for performing a security analysis of the UMLsec models against the security requirements that are included [14] and has been used in a variety of industrial projects [3]. The UMLsec extension is given in form of a UML profile using the standard UML extension mechanisms. Stereotypes are used together with tags to formulate the security requirements and assumptions. Constraints give criteria that determine whether the requirements are met by the system design, by referring to a precise semantics of the used fragment of UML. The security-relevant information added using stereotypes includes security assumptions on the physical level of the system, security requirements related to the secure handling and communication of data, and security policies that system parts are supposed to obey. The UMLsec tool-support can be used to check the constraints associated with UMLsec stereotypes mechanically, based on XMI output of the diagrams from the UML drawing tool in use. There is also a framework for implementing verification routines for the constraints associated with the UMLsec stereotypes. Thus advanced users of the UMLsec approach can use this framework to implement verification routines for the constraints of self-defined stereotypes. The semantics for the fragment of UML used for UMLsec is defined using so-called UML Machines, which is a kind of state machine which is equipped with UML-type communication mechanisms. On this basis, important security requirements such as secrecy, integrity, authenticity, and secure information flow are defined.

Applications including secure databases

Since this approach is tailored to the application, one can add the required level of security using formal proofs when necessary. Specialized operating system and hardware are also possible and may be needed to reach the required level of security. High-security systems require faithful application of basic security principles; for example, multilevel databases apply complete mediation. Databases work through transactions and a concurrency control system serializes transactions to prevent inconsistencies. High-security multilevel databases also require that the concurrency control system preserves security. The methods described in the last section still apply here, except that additional requirements must be considered. Because of this, these approaches are discussed in less detail, describing only two recent papers that contain references to past work.

Designing Secure Databases using OCL
An approach to designing the content of a security-critical data base uses the Object Constraint

Language (OCL) which is an optional part of the Unified Modeling Language (UML). More specifically, [8] presents the Object Security Constraint Language V.2. (OSCL2), which is based in OCL. This OCL extension can be used to incorporate security information and constraints in a Platform Independent Model (PIM) given as a UML class model. The information from the PIM is then translated into a Platform Specific Model (PSM) given as a multilevel relational model. This can then be implemented in a particular Database Management System (DBMS), such as Oracle9*i* Label Security. These transformations can be done automatically or semi-automatically using OSCL2 compilers. Related to this, [9] presents a methodology that consists of four stages: requirements gathering; database analysis; multilevel relational logical design; and specific logical design. Here, the first three stages define activities to analyze and design a secure database. The last stage consists of activities that adapt the general secure data model to one of the most popular secure database management systems: Oracle*9i* Label Security. They later extended the approach to data warehouses, multidimensional databases, and on-line analytical processing applications.

In both cases, a particular multilevel database system, meaning a set of users organized in levels, compartments, and groups, is given access to specific items of a relational database, according to the characteristics of those items, which also include levels, compartments, and groups. A set of rules describes the allowed access of users to data items. The secure metamodel is stored in the labels of each row or user definition. As indicated, the extra requirements can be superimposed in a general secure software development methodology.

Other approaches to Secure Software Development with Applicability to Databases

There are other approaches to developing security-critical software which can be applied to developing secure databases and database management systems.

[12] presents an approach for the predicative specification of user rights in the context of an object oriented use case driven development process. It extends the specification of methods by a permission section describing the right of some actor to call the method of an object. The syntactic and semantic framework is first-order logic with a built-in notion of objects and classes provided with an algebraic semantics. The approach can be realized in OCL.

[1] presents an approach to building secure systems where designers specify system models along with their security requirements and use tools to automatically generate system architectures from the models, including complete, configured access control infrastructures. It includes a combination of UML-based modeling languages with a security modeling language for formalizing access control requirements.

[15] presents an approach based on the high-level concepts and modeling activities of the secure Tropos methodology and enriched with low level security-engineering ontology and models derived from the UMLsec approach.

System architecture for security

Whichever approach is used, there are basically three general architectural configurations to include security functions:

- Figure 1 shows the standard approach. Here the DBMS and the operating system have their own set of security services.

- Figure 2 shows a way to unify the design of the DBMS with the design of the OS, using an I/O and file subsystem and a security subsystem to be used by both the DBMS and the OS.
- Figure 3 is an extension of the standard approach where a Web Application Server (WAS) unifies security for several databases. The WAS applies a common conceptual model to the information and can integrate different types of databases.

These configurations can be used in either of the approaches discussed earlier. Within each configuration it is possible to use security kernels and virtual machines.

KEY APPLICATIONS

Clearly, the first approach makes sense when the objective is a secure DBMS product, since it is not possible to know what user applications will be supported in the future. The only choice is then to build a system which is as secure as possible within these constraints and within a reasonable cost.

In the second case, the type of application to be supported is known. This gives the designers the flexibility of choosing an appropriate existing database system, as done in [9], or to build the DBMS to reach the required degree of security. If the complete DBMS is to be built, the first approach is appropriate, using as parameter the degree of security.

CROSS REFERENCES

Authorization, DBMS, DB Middleware, Distributed DBMS, Process structure, 00005, 00024, 00032, 00076, 00111, 00117, 00135, 00214,00249, 00251, 00274, 00320, 00332, 00333, 00641, 00659, 00665, 00701, 00703, 00810, 00825, 00898

RECOMMENDED READING

[10] is a good survey of this topic. While its references are old, the concepts discussed there are still valid. In particular, both approaches are well discussed. [8, 9] have good overviews (with recent references) of the application-based approach, with emphasis on the logical design of the schema, not on the software development aspects. For the general approach, the reader should consult the corresponding methodology descriptions, such as [7, 12, 14]. Reference [2] provides a good general survey of database security.

REFERENCES

[1] D.A. Basin, J. Doser, T. Lodderstedt: Model driven security: From UML models to access control infrastructures. ACM Trans. Softw. Eng. Methodol. 15(1): 39-91 (2006)

[2] E. Bertino and R. Sandhu: Database Security--Concepts, approaches, and challenges. IEEE Trans. Dependable Sec. Comput. Vol. 2, No 1, 2-19, 2005.

[3] B. Best, J. Jurjens and B. Nuseibeh, *Model-based Security Engineering of Distributed Information Systems using UMLsec,* 29th International Conference on Software Engineering (ICSE 2007), ACM, 2007, pages 581--590

[4] S. Castano, M. Fugini, G. Martella, and P. Samarati, *Database security*, Addison-Wesley, 1994.

[5] E. B. Fernandez, R. C. Summers, C. Wood, *Database Security and Integrity*, Addison-Wesley, Reading, Massachusetts, Systems Programming Series, February 1981.

[6] E. B. Fernandez, E. Gudes, and H. Song, ``A model for evaluation and administration of security in object-oriented databases", *IEEE Trans. on Knowledge and Database Eng.*, vol. 6, no. 2, April 1994, 275--292.

[7] E. B. Fernandez, M.M. Larrondo-Petrie, T. Sorgente, and M. VanHilst, "A methodology to develop secure systems using patterns", Chapter 5 in "*Integrating security and software engineering: Advances and future vision",* H. Mouratidis and P. Giorgini (Eds.), IDEA Press, 2006, 107-126

[8] E. Fernández-Medina, M. Piattini: Extending OCL for Secure Database Development. UML 2004: 380-394

[9] E. Fernández-Medina, M. Piattini: Designing secure databases. Information & Software Technology 47(7): 463-477 (2005)

[10] M. Fugini, “Secure database development methodologies”, in *Database Security: Status and Prospects,* C. E. Landwehr (Ed.), Elsevier Science Publishers B.V. (North- Holland, 103-129.

[11] X. Ge, F. Polack, and R. Laleau, Secure Databases: an Analysis of Clark-Wilson Model in a Database Environment, Advanced Information Systems Engineering - 16th International Conference, CAiSE 2004, Riga, Latvia, June 7-11, 2004, Persson A., Stirna J. (Eds.), LNCS vol 3084, 234-247.

[12] M. HafnerM. Alam[12] M. Hafner[HAB06] M. Hafner , [12] M. HafnerMoDELS 2006[12] M. Hafner[HAB06] M. Hafner , , R.Breu: Towards a MOF/QVT-Based Domain Architecture for Model Driven Security. [12] M. Hafner

[13] J. Jurjens, *Secure Systems Development with UML,* Springer-Verlag, 2004.

[14] J. Jurjens, “Sound Methods and Effective Tools for Model-based Security Engineering with UML”, *27th International Conference on Software Engineering (ICSE 2005),* ACM, 2005, pages 322--331.

[15] H. Mouratidis, J. Jürjens, J. Fox: Towards a Comprehensive Framework for Secure Systems Development. CAiSE 2006: 48-62

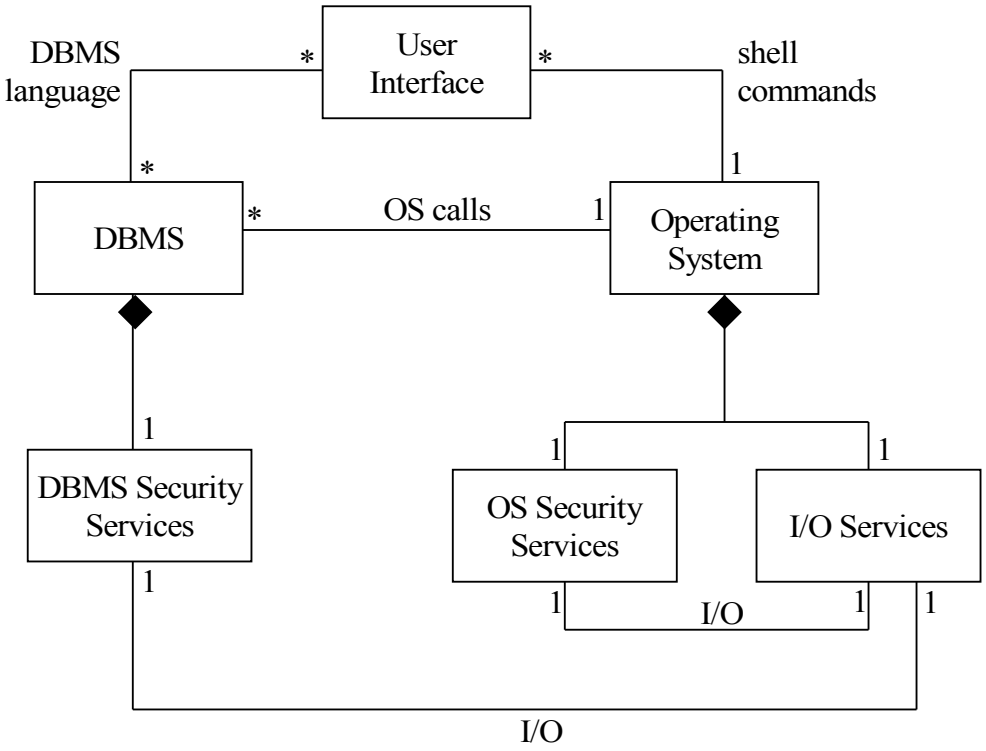

Figure 1. Standard placement fo security services

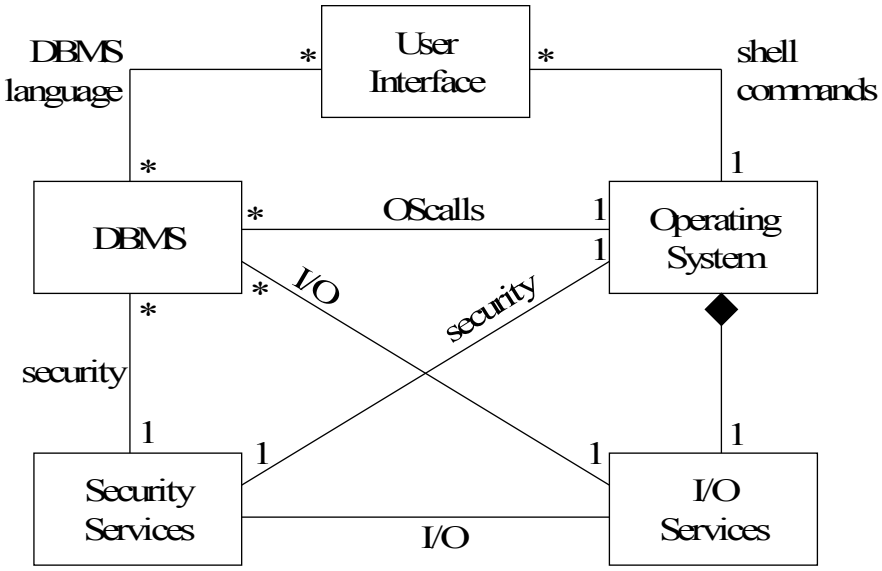

Figure 2. Common security services

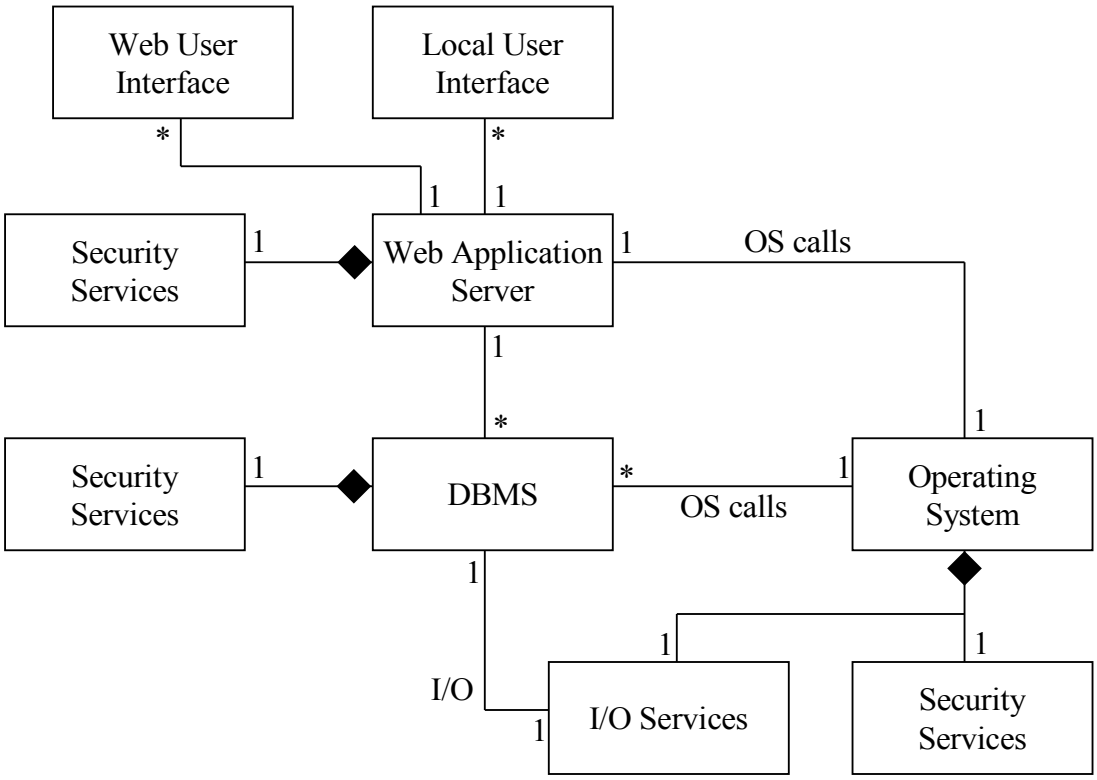

Figure 3. Architecture using a Web Application Server