A MODEL FOR FINE-GRAINED ACCESS CONTROL TO WEB DATABASES: VAMOS – A CASE STUDY

Peter Trommler
Georg Simon Ohm University of Applied Sciences Nuremberg
Kesslerplatz 12, 90489 Nuremberg, Germany

ABSTRACT

Flaws in a Web application’s access control code could expose core business data stored in databases to the Internet. In this paper we present VAMOS, an ambient assisted living project for seniors, as a case study to verify the validity of an access control model that is based on the notion of the user’s “own data.” The implementation is based on the concept of parameterized views in a database. The paper develops the “own data” model based on navigation through associations and temporal aspects representing actions in the past. A textual representation of the abstract model is presented and examples taken from VAMOS are displayed and discussed.

KEYWORDS

Web security, access control, databases, parameterized views, case study, ambient assisted living

1. INTRODUCTION

Databases are an important component in almost all Web applications, as they link the Web application to business functions. A Web application’s access control code is part of Web application security and flaws in that code could expose core business data to the Internet.

Access control in Web applications can be handled at various levels. Customers will be granted access to a certain subset of the functions of a Web application whereas a customer relationship manager will be allowed to see a different set of functions. These restrictions are normally implemented by a Web application server using access control based on URL prefixes. Ideally, each user is granted access to the subset of URLs required to carry out all permitted tasks following the principle of least privilege. This mechanism is coarse-grained as it does not place any restrictions onto access to the data that is accessed to carry out the business function. Fine-grained access control is considered to be part of the business function and is carried out as part of the implementation of the respective business functions. Thus, enforcement of access control at the data level relies on the correct implementation of all access control aspects of each business function in a Web application. Since that access control is naturally distributed across the code of the entire Web application, verification of its correctness involves a code review of the entire Web application code.

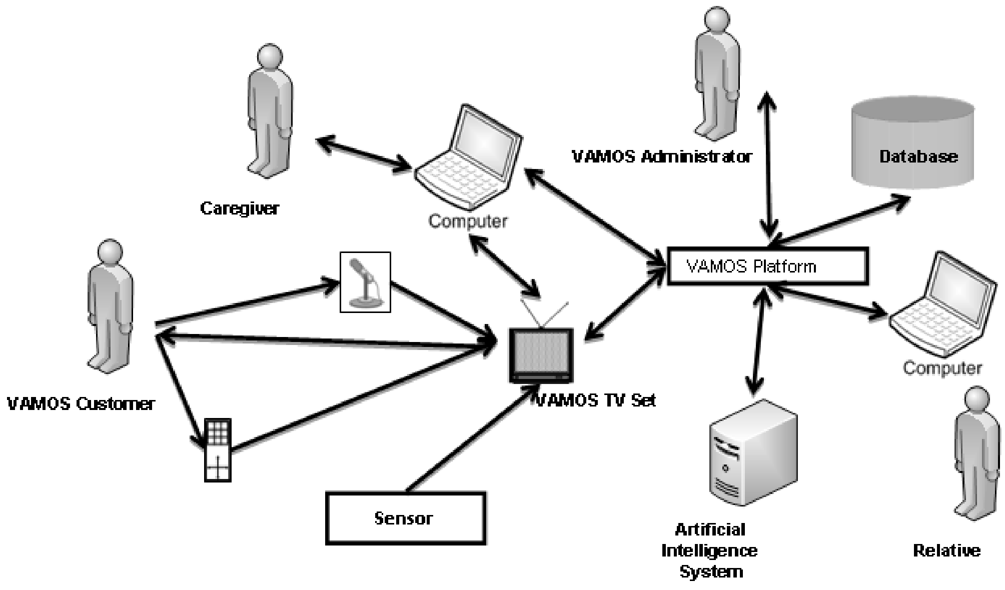
To cope with the issue of incorrectly implemented access control in Web applications, Roichman and Gudes (2007) proposed *Parameterized Views* as a mechanism to protect back-end databases from errors in Web applications. Views can be used to restrict access to database tables at the application level. Parameterized views can be used to pass information about the Web application user all the way down to the database, where access control decisions can now be based on that user’s privileges rather than the general access required by the Web application to cater to all users.

In this paper we study the nature of parameterized views in the context of the smart living environment VAMOS. We present a simple yet powerful model to specify access control rules and discuss the applicability of our model with regard to VAMOS’ access control requirements. In Section 2 we present an overview of the VAMOS system focusing on access control issues. Section 3 is devoted to the specification of access control rules in VAMOS. Section 4 gives pointers to related work, and Section 5 concludes the paper and lays out further research directions.

2. PROJECT VAMOS

VAMOS (**V**ersorgungseffizienz durch **a**ssistive, **mo**dulare Technologien in bedarfsorientierten **S**zenarien) is a research project in the area of ambient assisted living (Konnikov et al, 2011). Its goal is to provide technical assistance to elderly people who suffer from various limitations and who would need to move to a retirement home without VAMOS's support. Each VAMOS installation VAMOS can be tailored to the individual needs of the VAMOS user, and modules can be added and removed as needed. The individual configuration of each apartment is managed by a VAMOS administrator. Figure 1 displays an overview of VAMOS.

Figure 1. VAMOS Overview



The central component of every VAMOS installation is an Internet-capable television set that acts as a communication unit. The TV set displays information to the VAMOS Customer, collects information provided by sensors in the apartment (e.g. temperature, humidity) and on the VAMOS Customer (e.g. blood pressure, weight). Sensor data is communicated to a central server (the VAMOS Platform), where it is processed by artificial intelligence algorithms and recommendations are sent back to the VAMOS Customer.

Caregivers can add information to the VAMOS Customer records that will then be used to enhance the quality of recommendations for that user and which also serve as a means of quality assurance of the service provided by the caregiver. In addition, relatives of a VAMOS Customer can view the customer's health status, and caregiver reports, dependent on prior approval by the VAMOS Customer.

From the point of view of access control, six categories of principals can be identified (Prijovic and Trommler 2011): the VAMOS Customer, the VAMOS TV Set, caregivers, relatives, the VAMOS Administrator, and the Artificial Intelligence System. In the following sections we will briefly review the access control requirements for each of those principal categories.

2.1 The VAMOS Customer

A VAMOS Customer subscribes to VAMOS and uses a certain set of services and modules. VAMOS also offers services for communication amongst VAMOS Customers, provides information on activities in the neighborhood, and an electronic personal agenda and address book. VAMOS offers VAMOS Customers read access to sensor data and recommendations based on this data. This information is accessed through the user interface of a VAMOS TV Set.

Each service needs access to service-specific data stored in the database. A VAMOS Customer is allowed to access service-specific data about her own services but not data of other VAMOS Customers using the same service.

2.2 The VAMOS TV Set

A VAMOS TV Set is the central hub for communication with the VAMOS Platform and the VAMOS database. One or more VAMOS Customers use the VAMOS TV Set to access VAMOS services and other content on the Internet through a browser. In addition to serving as a "large display", a VAMOS TV Set communicates with sensors in the apartment, reads data from medical devices (e.g. weight, blood pressure), and sends commands to actors in the apartment (e.g. turn down the heaters).

To carry out these tasks a VAMOS TV Set will need to write sensor data from its sensors to the database. It must not, however, be allowed to access sensor data that belongs to sensors not attached to that particular VAMOS TV Set.

2.3 Caregivers

Caregivers maintain a log book of their activities on the VAMOS Platform. VAMOS Customers, the Artificial Intelligence System, and Relatives can view part of that information and, in the case of the Artificial Intelligence System, can derive new information and store it in the database. Caregivers will only be allowed to view and modify their own customers' care-related data.

2.4 Relatives

In a modern world where employee flexibility is important, relatives often live far away from a VAMOS Customer. Access via the Internet can be granted to Relatives to keep them informed of the general status of the VAMOS Customer. Since this concerns the VAMOS Customer's sensitive personal data, access will only be granted with the VAMOS Customer's consent. The VAMOS Customer can decide what level of access will be given to each relative, e.g. one relative might only be allowed to see general information on the wellbeing while another relative will be granted access to the caregiver's log book. Relatives can only view data on VAMOS Customers that have established them as a Relative and only to the degree that the VAMOS Customer has specified. All other data cannot be accessed.

2.5 The VAMOS Administrator

A VAMOS Administrator is responsible for the management of VAMOS Customers, apartments, and service subscriptions. She also manages entries for Relatives and Caregivers in the VAMOS Platform. The VAMOS Administrator is not a database administrator and hence access to the database can and must be restricted to the tables necessary to carry out management tasks.

2.6 The Artificial Intelligence System

VAMOS uses an Artificial Intelligence System to give recommendations to VAMOS Customers. These recommendations are based on sensor data and caregiver log book entries. The Artificial Intelligence System does not need to know the names of the VAMOS Customers and hence must not get access to the name field in a VAMOS Customer's record. It will, however, need access to other fields of a VAMOS Customer's record to link data from various sources (e.g. which sensor data records belong to sensors installed in the VAMOS Customer's apartment) to that particular VAMOS Customer. Read access to sensor data, the caregiver's log book, and the services subscribed to by a VAMOS Customer is granted, as well as write access to the table that stores recommendations.

2.7 Access Patterns in VAMOS

Looking at the various user categories of VAMOS three general patterns can be observed:

1. Access to a subset of database tables and a subset of the columns in those tables,
2. Access to the user's "own data" that can be found in a subset of tables and columns, but is additionally restricted to a subset of the records, and
3. Access to data is granted (or denied) based on actions in the past.

VAMOS Administrators and the Artificial Intelligence System require access modeled by pattern 1. All other principals need access to their "own data" according to pattern 2. Pattern 3 is required only for access by Relatives, based on actions in the past (e.g. the VAMOS Customer has previously granted access). A situation where actions in the past do not yet exist is interpreted as though no actions have taken place in the past. This is similar to the negation-as-failure rule presented by Clark (1978).

3. SPECIFICATION OF ACCESS CONTROL

The previous section showed that access control for each VAMOS user can be modeled by rules following three patterns. In this section we will develop a specification language for access control rules, demonstrate this language on examples taken from VAMOS, and present a mechanism based on parameterized views to enforce those rules.

3.1 Access Control Model

The description of our security model follows a meta-model for access control presented by Barker (2009). Our access control model can be described using the following four sets:

- Categories
- Principals: Who is asking for access?
- Resources: Which records can be accessed?
- Actions: What operations are permitted on an object?

In our model we do not require "situational identifiers" or "event identifiers" as defined in Barker's meta-model.

Having identified the above sets, an access control policy is defined as relations on these sets. The relation of Principals and Categories (*PCA*) and the relation of Principals, Resources and Actions (*PAR*) are required for our access control model.

Principals can be assigned to multiple Categories in general, which is reflected by the definition of a relation $PCA \subseteq P \times C$. In VAMOS each principal belongs to exactly one category and hence PCA can be defined as a mapping function from principals to categories.

The relation $PAR \subseteq P \times A \times R$ denotes all tuples of Principals, Actions and Resources that are permitted by the access control policy. To define that relation we use a two step approach. First, we define which resources can be accessed and second, we define which actions can be done with those resources.

3.1.1 Modeling Resource Access

A specification language for access control rules must include the following elements:

- View,
- Navigation,
- Traces, and
- Anchors.

A view specifies which operations are permitted per table. In the case of rules following pattern 1, a view can be directly translated into a database.

In an entity relationship diagram, associations link tables and navigation is used in UML diagrams to specify the direction in which a link can be used. To specify a principal's "own data", navigation is defined on associations on a per-category basis. Access is then granted to a record in a table if it can be reached through navigation from other accessible records. Views on each table serve to further restrict access to certain columns in the same way ordinary database views do.

Traces record actions in the past. In the case of the creation of a record in the database, that record serves as the trace. In an e-learning system, for example, students will only be allowed to view the sample solution to an assignment once they have submitted their answer. So, the record containing the answer acts as a trace. Once the student has had access to the sample solution, however, re-submission of an answer is no longer permitted. Unlike the create operation, a read operation does not leave a “natural” trace, so the fact that the student has already read the sample solution must be recorded in the database by creating a trace record. The trace record is not part of the business model of the database but must be created as part of the implementation of the access control system.

The definition of *own data* is an inductive definition: a set of records of *own data* in some table determines the set of records in another table through navigation. The starting point of this navigation is called an anchor. An anchor is to be found in a specific table which does not have to be identical for all principal categories. However, most applications choose to implement a “partner” table that contains basic information on all principals. In VAMOS, however, VAMOS TV sets are managed in a table separate from the other (human) principals. It is the responsibility of the authentication system to map an authenticated principal to a system-wide unique ID. Determining the principal category for an authenticated principal is performed as part of authorization, which may utilize data stored in the database or in some other external system such as a directory service. Given the identifier of an authenticated principal and the principal category, our access control system determines the appropriate anchor.

3.1.2 Modeling Actions

The discussion in the previous section focused on which records could be accessed by a particular principal. This section elaborates the details concerning the actions on these records. Actions on database records are the classical four operation types: create, read, update, and delete. We must specify what it means to carry out these operations on the principal’s *own data*.

Delete on the principal’s own data is straightforward as this operation has no parameters other than the record itself.

Read involves checking which fields the principal is allowed to read and returning only those.

Update is more difficult. Updating fields that are not part of the definition of *own data* is straightforward. Updating a field that is part of the definition of own data, however, might change ownership of the record and hence must be governed by an access policy. If change of ownership is not permitted then this policy can be enforced by restricting the values that can be written to that field to the set of values that characterizes the principal’s own data. Consider a calendaring application that allows a user to maintain several different calendars. Moving a calendar item from one calendar to another owned by the same user requires a restriction on the set of calendars the item can be moved to and hence a restriction on calendar identifiers permissible for the update.

Create, like update, could assign record ownership to other principals and the same arguments as for update operations apply. The values of fields that are used to determine ownership of a record must be restricted.

3.2 Syntax of a Specification Language

The discussion above leads to the definition of a formal specification language for the specification of an access policy. Parameterized views will then be generated from such a specification.

A parameterized view must be specified for each table. The business analyst, however, talks about access from the perspective of a principal category. Hence, the user category will be used to structure a specification. The specification of a view consists of three components: anchor, navigation and access. The grammar below reflects this structure. The grammar is given in extended Backus-Naur form where items in quotes represent terminal symbols. Identifiers that refer to database items were left as non-terminals in the grammar to increase readability.

To determine the user category for an authenticated user, parts of authentication data, e.g. attributes in an LDAP directory, or information in the database itself can be used. The anchor object is a record in the database that identifies the user and authentication data must be mapped onto data found in that record.

```
View ::= Anchor Navigation Access
Anchor ::= “Anchor:” Tablename “=” Mapping “;”
Navigation ::= Navigationitem+ “;”
Navigationitem ::= Tablename “->” Tablename “VIA” Linkspec
Linkspec ::= Tablename “.” Columnname |
             Tablename “.” Columnname “<->”Tablename “.” Columnname |
             Linkspec “AND” Predicate
Access ::= Tablename “:” Accessspecifier+ “;” | View
```

Navigation could involve several tables and is specified by naming the source and destination table and a predicate that describes the link (link specification). In associations where at least one side is unique (one-to-one, one-to-many, many-to-one), the column is specified by the column name in the other side of the association. In many-to-many associations both columns in both tables need to be specified. The syntax is redundant here as table names are repeated. Test with business analysts will demonstrate whether this enhances readability. In addition, general predicates in SQL can be added to further restrict navigation.

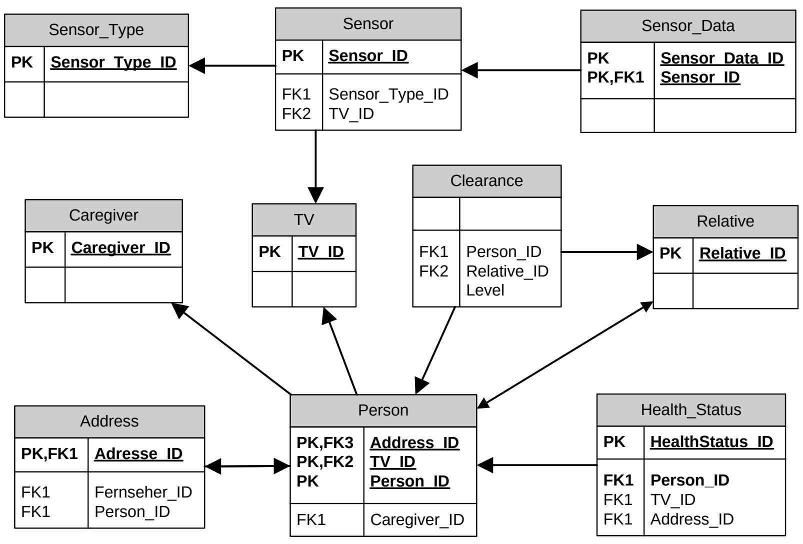
Finally, access to the table is specified in the form of a database view or using generic names for read, update, create, and delete operations.

3.3 Examples in VAMOS

To demonstrate the validity of our approach we will discuss three scenarios taken from the VAMOS project:

1. A VAMOS Customer reading her health status.
2. A VAMOS TV Set sending sensor data to the VAMOS Server
3. A Relative checking the health status of a VAMOS Customer

Figure 2. VAMOS database diagram (excerpt)



3.3.1 VAMOS Customer Reading Health Status

A VAMOS customer checks on her health status as recorded by the VAMOS Platform. To display the health status of a particular VAMOS Customer, read access to that customer’s data in table “Health_Status” is required. The records in table “Health_Status” will all contain a foreign key matching the primary key of the record representing the VAMOS Customer in table “Person.” The anchor is thus the record in table “Person” where the primary key matches the authenticated user. In the case of a VAMOS Customer we use the userid as a primary key so that the mapping between userid and primary key is the identity function. The access control rule can be specified as follows:

```
Anchor: Person.Person_ID = userid;
Person -> Health_Status VIA Health_Status.Person_ID;
Health_Status: read only;
```

3.3.2 VAMOS TV Set Sending Sensor Data

A VAMOS TV Set reads various sensors in the apartment and transfers that data to the VAMOS Server. Sensor data will then be added to table "SensorData" and never read, modified or deleted by the VAMOS TV. Thus, the only access right needed is permission to create a record in table "SensorData" whose foreign key matches one of the "Sensor_IDs" belonging to a sensor that is associated with that VAMOS TV Set. The anchor entity is the record of a VAMOS TV Set in table TV that matches the TV ID. Authentication of VAMOS TV Sets by is done by client authentication in TLS, which relies on public key certificates. An ID-function extracts the primary key of the VAMOS TV Sets from the certificate presented in the authentication process. The specification of a rule for storing sensor data is as follows:

```
Anchor: TV.TV_ID = ID(certificate);
TV -> Sensor VIA Sensor.TV_ID;
Sensor -> SensorData VIA SensorData.Sensor_ID;
SensorData: create;
```

3.3.3 Relative Checking Health Status of a VAMOS Customer

Our final example is a relative who has permission from the VAMOS Customer to check her health status. In general there is no restriction on how many relatives a VAMOS Customer can define and a relative can also have more than one VAMOS Customer. The many-to-many relation is reflected in the database model.

The VAMOS Customer can grant access to her personal data at various levels and can also revoke access for a particular Relative through the VAMOS Platform. This decision will be stored in the database. Here we have an example for a temporal restriction where the required action (access granted by VAMOS Customer) is recorded in the database and revocation of access is recorded by deletion of the respective record. In addition to our definition in Section 3.1, the access level must be checked to decide whether access to certain data should be granted. A specification of that access rule is:

```
Anchor: Relative = userid;
Relative -> Person VIA Person.relative <-> Relative.person
Person -> Health_Status VIA Health_Status.Person_ID
                     AND Clearance (person, relative, all);
Health_Status: read only;
```

4. RELATED WORK

Modeling security requirements for Web applications has been studied in the context of UML. UMLSec was presented by Jürjens (2005) as an extension to UML for the specification of secure systems in UML. UMLSec takes a general approach to security, where data is marked with confidentiality labels and those labels are used to determine cryptographic requirements for transmission of that data. Fine-grained access control could be specified as annotations but a formal language is not provided.

Basin et al (2006) propose SecureUML as the basis of "Model Driven Security." The goal of Model Driven Security is to generate security code. The language provides Roles, Permissions, Actions, Resources, and Authorizations Constraints. The latter could be used to specify access control rules as presented in this paper but again no formal way to do so is specified.

A proposed extension to the Business Process Modeling Notation (BPMN) to specify security requirements (Rodriguez et al, 2007) focuses on the specification of security requirements at the level of business analysts. The Privacy and Access Control elements come closest to what has been presented in this paper, and in those specifications both informal text and formal specifications stated in the Object Constraint Language (OCL) can be used. Our model is simpler that the full expressive power of OCL and hence more difficult for the business analyst to work with.

A survey of model-driven development platforms for Web applications by Valderas and Pelechano (2011) reviews access control models, among other features. As part of data requirements (DR), "accessing capabilities of users and system operations (DR3)" are considered. The models found in the surveyed platforms, however, focus on the restriction of user's Web navigation capabilities and hence a role-based access control model of specification is used. Per-user access control is considered in some models but then access control rules are specified explicitly rather than depending on database content.

Object Management Group (2001) describes a case study of a healthcare information system where the access control decision is based on static attributes, dynamic attributes, and "other factors." Since CORBA merely specifies interfaces of compliant middleware systems, no specification language has been defined.

An extension to Role-Based Access Control (Sandhu et al 1996) is privacy-aware RBAC (Ni et al 2007 and Ni et al 2008) that uses context variables to record privacy-relevant information. Traces used in our model are a restricted form of context variables as they merely record actions of principals.

5. CONCLUSION

In this paper a model and specification language for access control in Web application has been developed. The access control model is based on the notion of "own data." Access control is specified at the database level and hence the Web application code can be treated as only partially trusted. We presented VAMOS, an ambient assisted living system for seniors, as a case study, and validated the approach in that project.

The paper presented a textual language for specifications. A graphical notation, however, might be easier to understand for business analysts. Evaluations with business analysts will have to be carried out in the future and integration with UML and BPMN models will be studied.

The presented model could form a solid basis for the enhancement of modeling tools that are used for generating significant parts of Web applications. The modeling tools studied in Valderas et al (2011) could be a good starting point.

ACKNOWLEDGEMENT

This research was supported by the German Federal Ministry of Education and Research grant 16KT0943.

REFERENCES

Barker, S., 2009. The Next 700 Access Control Models or a Unifying Meta-Model? *In SACMAT'09: Proceedings of the 12th ACM symposium on Access control models and technologies*, Stresa Italy, pp. 187-196.

Basin D. et al, 2006. Model driven security: From UML models to access control infrastructures*. In ACM Transactions on Software Engineering Methodologies*, Vol. 15, No. 1, pp. 39-91.

Clark, K. L., 1978. Negation as Failure. In Logic and Data Bases, Vol. 1, pp. 293-322.

Jürjens, J., 2005. *Secure Systems Development with UML.* Springer Verlag, Heidelberg, Germany.

Konnikov A. et al, 2011. VAMOS – Assistenz nach Bedarf. *In Proceedings: Demographischer Wandel - Assistenzsysteme aus der Forschung in den Markt (AAL 2011)*, Berlin, Germany, CD-ROM file 74.

Ni, Q. et al, 2007. Conditional privacy-aware role based access control. *In ESORICS '07: Proceedings of the 12th European Symposium On Research In Computer Security*, pp. 72-89.

Ni, Q. et al, 2008. An Obligation Model Bridging Access Control Policies and Privacy Policies. *In SACMAT '08: Proceedings of the 13th ACM symposium on Access control models and technologies*, Estes Park, CO, USA, pp. 133-142.

Prijovic S. and Trommler P., 2011. Zugriffskontrolle in der Datenbank: Vamos eine Fallstudie. In DACH Security 2011, pp. 147-156.

Roichman A. and Gudes E., 2007. Fine-grained Access Control to Web Databases. *Proceedings of SACMAT'07,* Nice-Sophia Antipolis, France, pp. 31-40.

Sandhu, R. eta al, 1996. Role-based access control models. *IEEE Computer*, Vol. 29, No. 2, pp. 38-47.

Object Management Group, 2001, Resource Access Decision Facility Specification, Needham, MA, USA.

Valderas, P. and Pelechano V., 2011. A Survey of Requirements Specification in Model-Driven Development of Web Applications, *In ACM Transactions on the Web*, Vol. 5 No. 2, pp. 10:1-10:51.