Recovering Role-Based Access Control Security Models from Dynamic Web Applications

Manar H. Alalfi, James R. Cordy, and Thomas R. Dean

School of Computing, Queens University,
Kingston, Ontario, Canada
{alalfi,cordy,dean}@cs.queensu.ca

Abstract. Security of dynamic web applications is a serious issue. While Model Driven Architecture (MDA) techniques can be used to generate applications with given access control security properties, analysis of existing web applications is more problematic. In this paper we present a model transformation technique to automatically construct a role-based access control (RBAC) security model of dynamic web applications from previously recovered structural and behavioral models. The SecureUML model generated by this technique can be used to check for security properties of the original application. We demonstrate our approach by constructing an RBAC security model of PhpBB, a popular internet bulletin board system.

1 Introduction

Models provide a formal basis to specify various properties of software, such as access control properties. When the application is later generated from the model, developers have a reasonable expectation that these properties will be implemented in the software. However, determining the access control properties of an existing software application is a non-trivial task. We can try to verify the properties directly on the source code, or we can recover a model from the code that is amenable to analysis. One particular area of interest is dynamic web applications, which are often designed to interact with the general public and thus are directly accessible to a wide variety of attackers. In many current web applications, access control policies are spread throughout the application, making understanding and maintaining them a difficult task [1].

Security and vulnerability analysis of dynamic web applications is not new. Pistoia et al. [2] survey a variety of techniques that check for vulnerabilities such as SQL injection and cross site scripting. Alafi et al. [3] present a comprehensive survey of models and methods for web application verification and testing. They found that while models were built to analyze static and dynamic properties of the system, none of the surveyed techniques were able to model or check the access control policies of dynamic web applications.

In this paper we use TXL [4], a source transformation tool, to transform previously recovered structural and behavioral models [5, 6, 7, 8] to a role-based access control (RBAC) [9] model. The target model, a SecureUML [10] model expressed in XMI 2.1, can then be used to check that the desired access control properties are correctly implemented in the code. TXL is a source transformation language originally used at the source code level, but recently shown to be useful in transforming models [11, 12]. Such transformations are applicable to large models, including heterogeneous models that integrate components in a variety of languages. Using source transformation techniques allows us to integrate diverse models, to map platform-independent models to platform-specific ones, and to handle other tasks that involve dealing with multiple meta-models at once.

The key contributions of this paper are:

– An approach and tool to automatically recover security models from dynamic web applications using source transformation technology.
– A demonstration of the approach to recover a role-based security model from a widely used real world web application, PhpBB.

This paper is organized as follows: Section 2 introduces PhpBB as a running example used to demonstrate our technique and explains why it is an appropriate choice. We give an overview of our reverse engineering approach to recovering security models from a dynamic web applications in Section 3. The construction of a SecureUML security model from recovered structural and behavioral models is presented in Section 4. Section 5 highlights the advantages of the transformation-based approach in considering the correctness and completeness of the recovered models. Section 6 reviews related work, and Section 7 concludes the paper and presents directions for future work.

2 Running Example

We demonstrate our technique on PhpBB 2.0 [13], an internet bulletin board system used as a running example throughout the paper. Our focus is on recovering security models from production web applications, whose recovered models are much too large to show in a paper. Hence we show only snippets of the recovered models and concentrate on only three roles to illustrate our approach. In practice our method efficiently recovers complete RBAC models for multiple roles from production dynamic web applications of any size.

Our present implementation, or rather the front end that recovers the structural and behavioral models, is designed to handle web applications built using Apache, PHP and MySQL. Our choice of this combination is based on the popularity and predominance of these technologies on the web [14, 15, 16]. While we have thus far concentrated on these technologies, our overall approach is not technology dependent and can be extended to other choices as well. The security model construction approach of this paper is also not limited to automatically recovered models. It can be used to construct security models from any structural and behavioral models that conform to the meta-models provided in Figure 3, including those crafted by hand or using MDA authoring tools.

3 Overview

Figure 1 shows the general framework of our technique to convert the structural and behavioral models into a SecureUML security model. The work described in this paper is part of a larger toolset to analyze role-based access control which begins with automated recovery of structural and behavioral models described in detail elsewhere [5, 6, 7, 8]. The lower left (SQL2XMI) represents our automated recovery of the structural model (represented by an ER Data Model) from the application's schema source, while the upper left (PHP2XMI, WAFA, DWASTIC) represents the automated recovery of the application's behavioral model (represented by a sequence diagram) using a combination of static and dynamic analysis. In this section we give a brief overview of the reverse engineering of these models, which is described in full in other papers.

The remainder of Figure 1 describes the model transformation process which is the subject of this paper. We begin by building a dictionary of of the entities that will form the core of the security model. We use this dictionary of entities to identify the attributes, relations and constrained events affecting them in the recovered models. These are then mapped to a SecureUML security model, an example of which is shown in Figure 2. This transformation process is presented in detail in Section 4.

The model recovery process begins with a static analysis to recover a structural model of the application resources as they pertain to the user's use of systems, applications and business processes. While UML is considered the standard for application modelling, there is no corresponding open standard for data modeling. SQL2XMI [6] is the technique we use to automatically transform an SQL DDL schema to a UML 2.1 ER diagram. The top part of Figure 3 shows the meta-model for the recovered UML-ER data model representing the structure of the application.

The second part of the model recovery uses a combination of static and dynamic analysis to recover a behavioral model of the application (Figure 3). A set of three tools, PHP2XMI [5], WAFA [8] and DWASTIC [7] is used to recover this model. First, PHP2XMI uses source transformation to instrument the

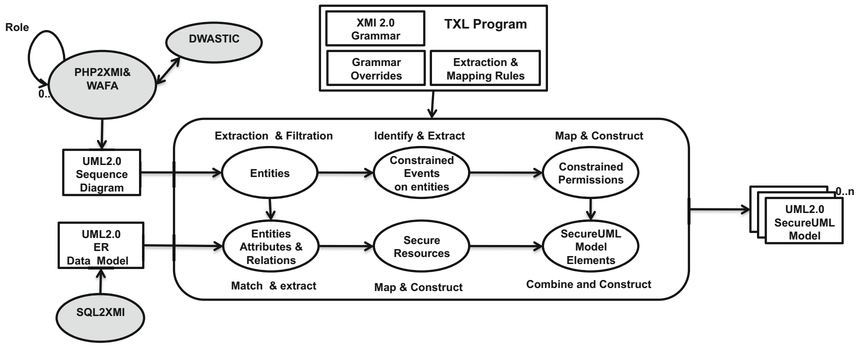

Fig. 1. PHP2SecureUML Framework

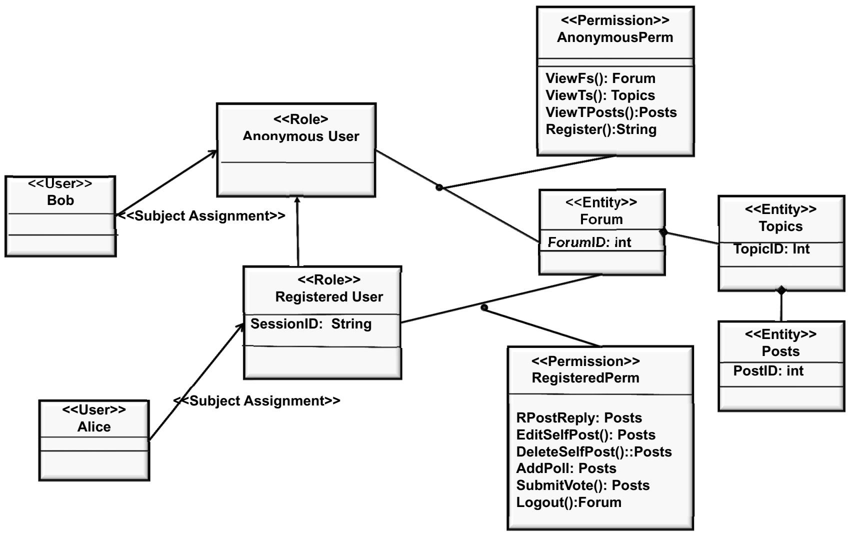

Fig. 2. An example SecureUML model

code and exercises it to recover the permissions associated with each user role, representing the result as a UML sequence diagram.

This sequence diagram is then extended by WAFA, which recovers a fine-grained interaction model from the application. From the point of view of user interaction, the secured resources are represented by lifelines for the web pages delivered by the application. The diagrams operations are used to map the different type of access allowable to the user over the applications secured resources. All aspects of this access are captured as either operations parameters or constraints. These include the accesss type, timestamp, condition, return value, and unique id. The unique id is to identify the accesss relation with the source code and the source page. Message names encode a combination of these values in a single string. The lower part of Figure 3 shows the recovered elements (shaded) based on the UML sequence diagram meta-model. The black dashed line represents the relation between the recovered sequence and ER diagrams, where each entity in the ER diagram (secure resource) is mapped to a class in the sequence diagram represented as a lifeline.

Because our behavioral model recovery uses dynamic analysis to explore the behavior of the application, a measure of completeness for the recovered behavioral model is required. For this purpose, we have developed DWASTIC, a tool that augments the dynamic analysis with additional code coverage instrumentation. DWASTIC uses several coverage criteria specialized for web applications to help ensure that all pages and potential interactions are explored. It provides a direct way to trace those parts of code that are not covered by test cases, and serves as a coverage measure for extraction of the access control security model. The accuracy of the results is both hand verified and robust since it is automatically back-checked at run time.

4 SecureUML Model Construction

In the previous section we briefly outlined how SQL2XMI, WAFA and PHP2XMI help us to recover UML-based structural and behavioral models from web applications, and how DWASTIC provides a measure of coverage. In this work we use the relevant elements of these two recovered models to construct a role-based (RBAC) security model that conforms to the SecureUML [10] meta-model (middle of Figure 3).

SecureUML is an implementation of the Model Driven Security approach, a specialization of Model Driven Architecture. It explicitly integrates security aspects into the application's models and provides support for model transformation. The approach has been proposed to bridge the representation gap between the graphical languages used for specifying application design models,

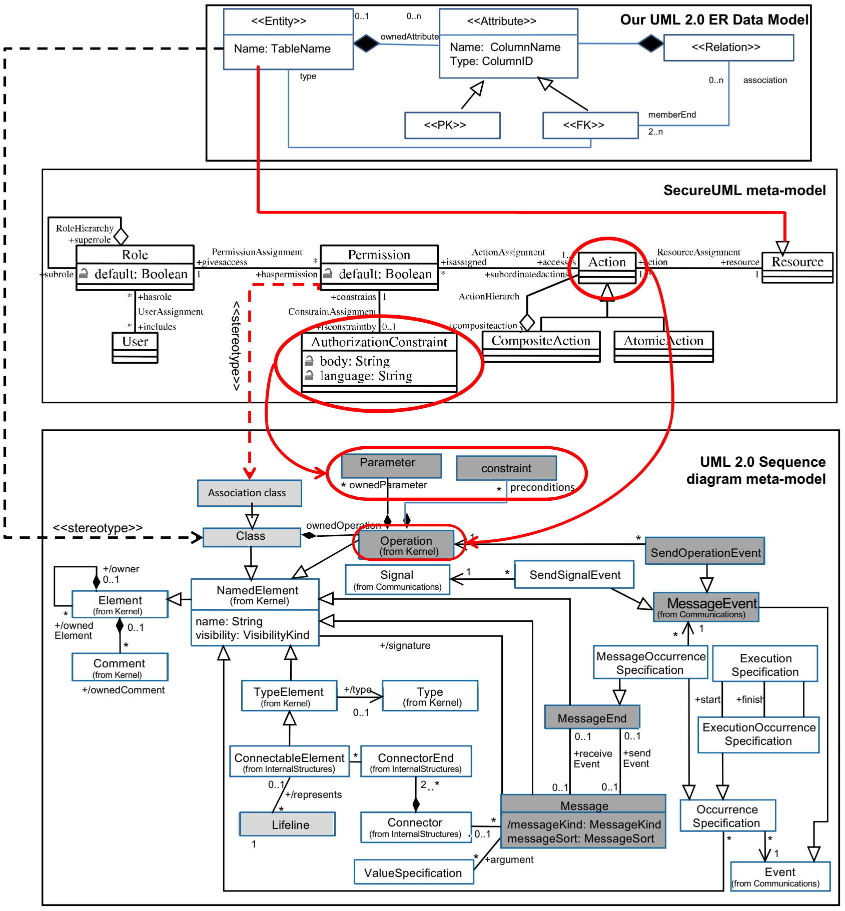

Fig. 3. UML2.0 Structural and behavioral meta-models and their mappings to the SecureUML [10] meta-model

such as UML, and the textual languages used to specify security models. It is built on a modular schema that comprises three basic elements: a language for security policy specification; a language for design model construction; and a dialect for defining integration points in these two languages. The abstract syntax for SecureUML is based on role-based access control (RBAC) [9]. It defines a meta-model that extends RBAC with authorization constraints to enable formal specification of access control policies that depend on dynamic aspects of the system, such as the access date or the values of the system's environment variables. The modeling notation for SecureUML is based on a UML profile that uses UML stereotypes and tagged values to represent the abstract syntax elements in the meta-model schema. Users, groups, and roles are represented as classes with stereotypes $\ll$ *User* $\gg$, $\ll$ *Group* $\gg$ and $\ll$ *Role* $\gg$ respectively, and permission is represented as an association class with a $\ll$ *Permission* $\gg$ stereotype.

Figure 2 shows an example of a SecureUML model for a web forum application. The diagram shows two users in different roles who are permitted different sets of actions based on their roles. Bob, who is an anonymous user, is permitted to access the forum entities using read operations. So, Bob can access a forum via *ViewForum()*, read a forums' topics via *ViewTopic()*, read topic posts via *ViewTPosts()*, and can register in a forum. Alice, who is a registered user of the forum, can not only perform all the operations available to Bob but is also permitted write access to the forum. Thus, she can also reply to posts via *RPostReply()*, edit her own posts using *EditSelfPost()*, and so on.

Our work defines a mapping from the recovered structural and behavioral meta-models to the target SecureUML meta-model which forms the basis of our transformation. Figure 3 shows the relationship between the UML-sequence diagram meta-model and our UML 2.0 data meta-model. The *Entity* element in the UML data meta-model corresponds to the *Class* element of the sequence diagram meta-model via a stereotype relationship, represented as a dotted line in the figure. Based on this relationship, the structural information for each entity in the sequence diagram can be pulled from the data model.

We implement this phase as a model transformation encoded as a sequence of source transformations in TXL [4]. Although the model transformation process accepts models as an input and generates models as output, where each of these conforms to a specific meta-model and reflects a particular view of the system, we can implement the transformation process between source and target models as a source-to-source transformation as long as they can be serialized into a text-based format. Fortunately, this can be easily done by most modeling tools, including ArgoUML and RSA, using the XMI export and import facility. While modeling tools often use different versions of XMI, TXL grammars can be adapted to accept and manipulate a range of XMI versions, and can generate multiple versions of the serialized models to match a range of modeling tools.

4.1 Entity Extraction and Filtration

The set of classes (entities) in the sequence diagram is the abstract representation of the diagram's lifelines and maps to the application's secure resources, which include the application server, browser session, and database-backend entities. In this first step these elements are identified and filtered to remove any redundancies, using source transformation.

We have developed a TXL grammar for XMI schemas which enables the manipulation of models that conform to the UML sequence diagram (SD), UML-based ER diagram and SecureUML meta-models. The process accepts as input a serialization of both the SD and ER models, and uses a rooted set of source transformation rules to enable the model's manipulation, integration and transformation to construct the target security model.

The transformation begins by searching for the set of secure resources that are engaged in the interaction behavior modeled by the SD. These elements are represented abstractly as a set of classes, and graphically as a set of lifelines. The corresponding source transformation rule (Figure 5) matches all SD class elements in the XMI representation and filters out any redundant ones. Redundancies can occur due to the fact that multiple secured resources receiving the same set of actions are represented as a single class and modeled using a single lifeline. The names of these resources are combined into a single string which represents the class name. Thus the transformation rule must refactor the combined string to identify the names of the corresponding secure resources.

Figure 4 presents a snippet of a recovered sequence diagram showing the results of the the first step of our approach, the list of entities engaged in the interaction. Some of the entities shown in the diagram snippet are: {*phbbb_forums,*

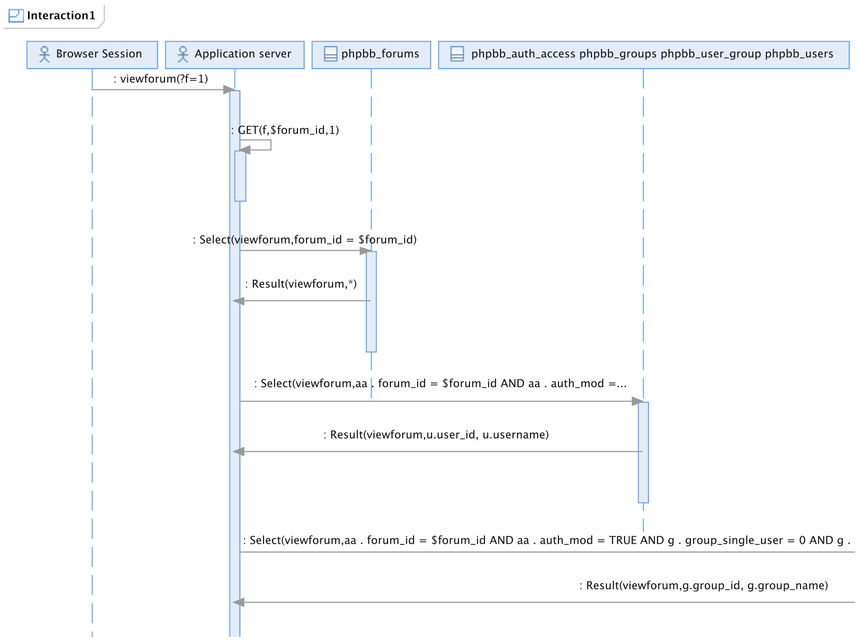
Interaction1
Browser Session
Application server
phpbb_forums
phpbb_auth_access phpbb_groups phpbb_user_group phpbb_users
: viewforum(?f=1)
: GET(f,$forum_id,1)
: Select(viewforum,forum_id = $forum_id)
: Result(viewforum,*)
: Select(viewforum,aa . forum_id = $forum_id AND aa . auth_mod =...
: Result(viewforum,u.user_id, u.username)
: Select(viewforum,aa . forum_id = $forum_id AND aa . auth_mod = TRUE AND g . group_single_user = 0 AND g .
: Result(viewforum,g.group_id, g.group_name)

Fig. 4. A snippet of the UML2.0 Entity-level sequence diagram for PhpBB 2.0 generated by WAFA and PHP2XMI

```
% Search for class elements in the sequence diagram
 deconstruct PackagedElement
    '< 'packagedElement 'xmi:type = XmiType [stringlit]
        'xmi: 'id =  ClassID [stringlit]
        'name = ClassName [stringlit] '>
        owndOp [repeat XMItoken]
    '</ 'packagedElement '>
 where
    XmiType [= "uml:Class"] [= "uml:Actor"]
```

Fig. 5. A small part of the TXL rule to extract class elements from the behavioral model sequence diagram. This pattern matches all Class and Actor elements in the XMI 2.0 representation of the recovered sequence diagram for the web application.

```
% Search the structural model for a matching Entity description
% for the EntityName extracted from the behvioural model
 match * [repeat XMItoken]
    '< 'packagedElement 'xmi:type = "uml:Class"
        'xmi: 'id =  EntityName  'name '= ClassName [stringlit] '>
        owndAttrib [repeat XMItoken]
    '</ 'packagedElement '>
    More [repeat XMItoken]
% Extract the entity's owned attributes
 construct OwnedAttribElements [repeat owned_Attribute]
    _  [^ owndAttrib]
% And its owned attribute relations
 construct  OwnedAttribRelElements [repeat owned_AttributeRel]
    _  [^ owndAttrib]
```

Fig. 6. A small part of the TXL rule to extract the attributes and relations associated with each entity from the structural ER data model. This pattern matches the elements in the XMI 2.0 representation of the recovered structural diagram for the web application corresponding to the entities extracted from the behavioral model.

phpbb_auth_access, phpbb_user_group, phpbb_users, Browser Sessions, Application Server}. Note that the set of entities representing the third lifeline has been re-factored into separate entities.

4.2 Entity Attribute and Relation Extraction

Once the set of secure resource elements engaged in interaction behavior has been identified, another source transformation rule is applied to each of the identified elements (Figure 6). This subrule consults the UML ER diagram to search for structural information relevant to those elements, including attributes and relations with other resources.

Conceptually, the transformation rule searches for all class elements with the Entity stereotype in the ER model that matches one of the entities identified in the previous section. It extracts the entities' attribute elements and associations with other entities in the identified set. The result of this phase is an ER diagram of the secure resources engaged in interactions in a particular browsing session.

In our running example, the set of entities extracted in the previous step is used to extract the entities' attributes and relations from the recovered ER diagram of the system, a snippet of which is shown in Figure 7. This step is necessary so that Entities, attributes and relations not relevant to the target

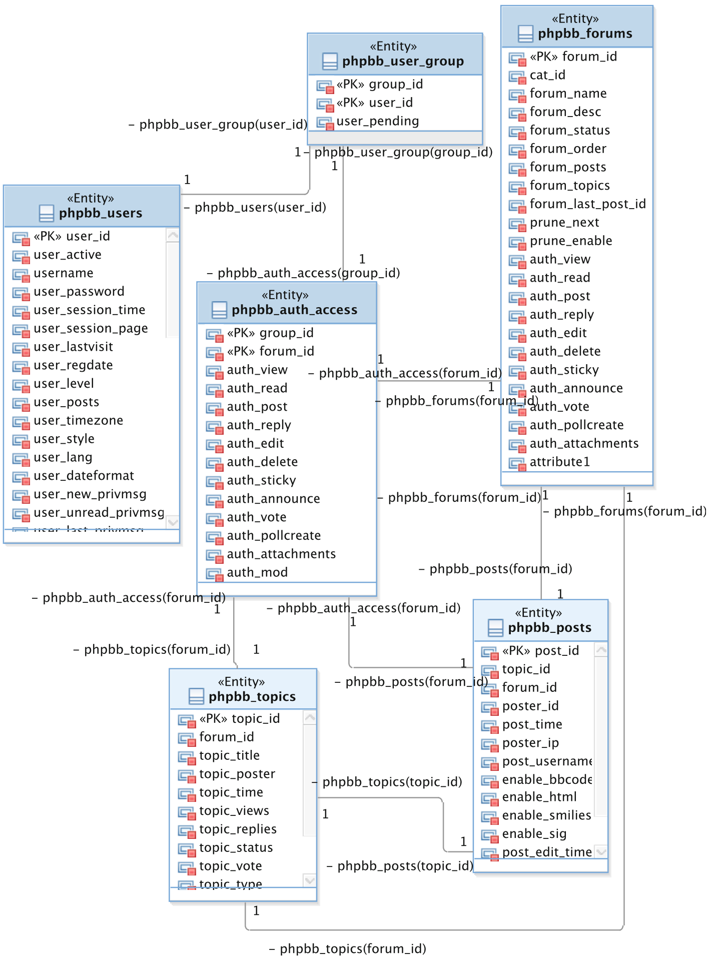

Fig. 7. A snippet of the recovered ER data model diagram for PhpBB 2.0

security model (i.e., not involved in the interactions) are filtered out and not extracted. Thus the $\{Topic, Post\}$ entities, in the ER diagram (Figure 7) and their attributes and relations will not be included in the artifacts used to construct the diagram representing the secure resources.

As an example, some of the attributes and relations extracted by this step for the *phpbb_forums* entity are: $\{<< PK >> forum\_id, cat\_id, forum\_name\}$, and the association attribute between *phpbb_forums* and *phpbb_auth_access*. Note that relations between *phpbb_forums* and *phpbb_ topics*, *phpbb_posts* are not recovered because they are not part of the interactions described in our example.

4.3 Constrained Event Extraction

The set of permissions allowed on each of the recovered entities (i.e., secure resources) is modeled as the message receive events of the corresponding lifeline. Each recipient event element in the sequence diagram meta-model is represented as an operation which may be associated with parameters and constraints. The next source transformation rule receives as a parameter the set of recovered resources, matches the elements of the serialized sequence diagram, and whenever a class with the same resource name is matched, identifies and extracts the set of all operation elements associated with the class, along with its parameters and constraints (Figure 8).

The rule then constructs the meta-model elements of SecureUML to represent the recovered permissions. Each operation element and its parameters is mapped to a *permission* action, and operation constraints are mapped to *authorization* constraints. The rule constructs an association class to represent the set of recovered operations for each specific resource. The association class is marked as a *permission* stereotype to reflect its security semantics.

For each entity in the resulting secure resources diagram of the previous step, the set of actions, action constraints and other relevant parameters are extracted. For instance, for the *phpbb_forum* entity, these artifacts are:

$Action : Select(allattributes).$
$Constraint(forum\_id = \$forum\_id).$
$Timestamp(1247604868).$
$ActionIdInCode(viewForum, 366).$

where *viewForum* is the page name and 366 the operation ID in the source code.

```
% Identify and extract operation elements corresponding to
% extracted secure resources
 deconstruct ownedOp
    '<'ownedRule 'xmi:type '= "uml:Constraint"
        'xmi:id '= ConsID [attvalue]
        'name '= AcName [attvalue]
        'constrainedElement '= ConstElm [attvalue] '>
    <'specification 'xmi:type '= "uml:OpaqueExpression"
        'xmi:id '= ConsExprID [attvalue]
        'name '= ConsExprName [attvalue] '/'>
    '</'ownedRule'>
    ownedOp2 [repeat XMItoken]
```

Fig. 8. A small part of the TXL rule to extract operation elements corresponding to identified resources from the recovered behavioral model sequence diagram. This pattern matches ownedRule elements in the XMI 2.0 representation of the recovered sequence diagram corresponding to each secure resource identified in the previous step.

4.4 SecureUML Model Element Construction

The previous steps have identified all the security elements necessary to construct the RBAC security model. In this step, we construct a security model that conforms to the SecureUML meta-model shown in Figure 3. A set of transformation rules is used to construct the security model, in which the extracted sequence

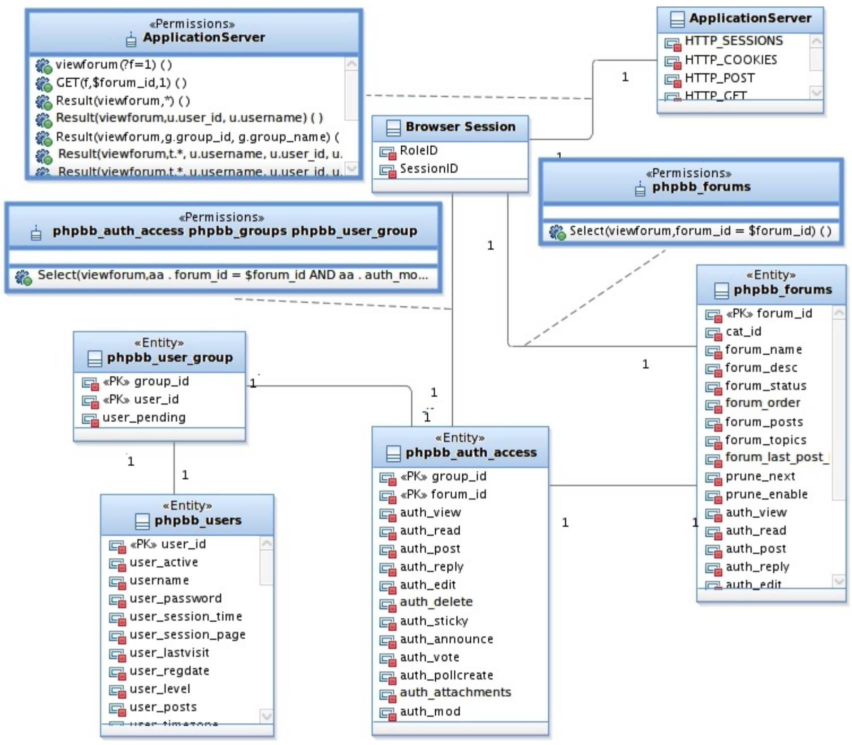

Fig. 9. An example generated SecureUML model instance for PhpBB 2.0

diagram's operations are mapped into permissions, operation constraints into authorization constraints, and the entities of the ER data model into resources.

In the SecureUML notation, the representation of resources is left open, so that developers can decide later which elements of the system they consider secure and to which they want to apply access constraints. These elements are defined using a dialect. In section 4.2, we identified the secure resources and represented them as an ER diagram. In section 4.3, we recovered the permission-action pairs and authorization constraints, and represented them as an association class with parameters and preconditions. Using a final TXL transformation rule, two association links are created: one that connects the association class, stereotyped by permissions, with the entity (resource) affected by the permission's actions; and a second that connects the acting role with the constructed association class.

In our running example, Figure 1 shows snippets of the resulting SecureUML model. For instance, an association class with name *phpbb_forum* is constructed with permission stereotype and attached to the entity *forum*, and the *browser session* entity represents the user role accessing the forum. The final result is a complete SecureUML model of the web application which can be checked for security properties using a standard model checker or custom analysis. In our case, the resulting SecureUML model is transformed once again into a Prolog formal model checked using Prolog rules to find potential access role violations.

5 Correctness and Completeness of the Recovered Model

One of the advantages of using a formal source transformation system for deriving and exploring security models from source code is that it is easier to reason about completeness and correctness of the tools. By contrast with hand-coded analyzers implemented in Java or C, source transformation rules can be tested and verified piecewise.

Because source transformations are based on parsing technology, the well-formedness of the results is guaranteed. TXL transformation rules are simply incapable of producing a result that does not conform to the syntactic forms of the target grammar/metamodel. The question of the semantic soundness of the constructed security model is also made simpler using a source transformation technique. Rather than having to reason about an entire hand-coded analysis program all at once, each TXL source transformation rule can be considered independently of the others. Whether the entire transformation is correct then becomes just a question of whether the set of rules forms a complete transformation, which can be checked separately. In our system this question is addressed by separating the process into a sequence of separate source transformation steps. Because each step yields a concrete intermediate text file representation that the next step parses as input, erroneous or incomplete results of a step are typically caught immediately by the next step. For example, if the data model extracted from the web application's schema is missing anything, there will be unresolved links when integrating the models that will make this fact immediately evident in the next transformation step.

Using source transformation rules to analyze the schema, source code and behavioral models also assists in guaranteeing completeness. For example, the TXL parser syntactically identifies all references to the SQL database in the source code, and the transformation rule for analyzing them simply transforms them to an instrumented form. The question of whether we have missed any database interactions in the extracted model is therefore easy to evaluate, simply by counting the number of SQL interactions in the model and comparing it to the number identified by the parser in the source. Dynamic behavioral completeness is handled by including coverage counters in the instrumentation, implemented using the DWASTIC tool discussed in section 3.

Our approach has been validated in a case study to extract RBAC security models from PhpBB 2.0. Table 1 presents some statistics on the size of the recovered models for two roles: anonymous (guest) user and registered user. The table presents results in terms of the number of resulting model elements as described in Section 2. These results are based on an ER diagram consisting of 30 entities, 370 attributes, and 55 relations, and a sequence diagram with a partial behavioural coverage of 50% of PhpBB 2.0 by page access and an average of %20 by back end access (to database and server environment variables). Even with

Table 1. Sizes of recovered SecureUML models for the anonymous and registered user roles in PhpBB 2.0

Anonymous (Guest User) Role

Entities	Attributes	Relations	Operations	Permissions	Constraints	Parameters
12	191	36	321	14	1465	9567

Registered User Role

Entities	Attributes	Relations	Operations	Permissions	Constraints	Parameters
15	249	62	59294	24	28304	177627

this relatively low percentage of coverage we can notice the variation in the size of the recovered SecureUML models, specifically for the number of operations, constraints and parameters, which reflects the fact that registered users have significantly more access than guest users.

A more detailed evaluation of the recovered models will be presented in [17], where the resulting models are transformed into a formal Prolog model and checked for security properties. We are also working on validating the approach using a larger case study, recovering RBAC security models for the educational support web application *Moodle*.

6 Related Work

Most of the early literature on web application security concentrates on the process of modeling the design of the web applications. It proposes forward engineering-based methods designed to simplify the process of building highly interactive web applications [18, 19, 20, 21]. Other research uses reverse engineering methods to extract models from existing web applications in order to support their maintenance and evolution [22, 23, 24], However, few approaches recover security models from web applications, and in particular access control security models. In Alalfi et al. [3] we survey the state of the art in these techniques.

The most relevant related approach in this domain is the Letarte and Merlo approach [25] which uses static analysis to extract a simple role model from PHP code, and more specifically from database statements. Changes in authorization level in the code are modeled using an inter-procedural control flow graph with three kinds of edges: positive-authorization (change to admin), negative-authorization (change to user), and generic (no change in security level). A predefined authorization pattern is used to identify transfer of control in the code and changes in authorization level in the extracted model. Unlike our approach, the Letarte and Merlo approach simplifies to only two roles (admin vs. user) for which access may or may not be granted to database statements. The model is based on an application-dependent authorization pattern and does not provide any link back to the source code.

Other approaches have been proposed to recover models and/or check for access control properties for domains other than web applications, with a focus on Java based applications. Koved et al. [26] use context sensitive data and control flow analysis to construct an Access Right Invocation graph, which represents the authorization model of the code. This enables identification of classes in each path that contain a call to the Java 2 security authorization subsystems. The approach is used to automatically compute the access rights requirements at each program point in mobile Java applications such as applets and servlets.

Pistoia et al. [2] statically construct a call graph to represent the flow of authorization in an application by over-approximating method calls and identifying access-restricted methods. The graph is used as the basis of several security analyses, including detecting if the application's RBAC security policy is restrictive or permissive. The authors generate reports on code locations that have inconsistencies and suggest a replacement policy to eliminate the vulnerabilities. The approach is implemented as a part of IBM's Enterprise Security Policy Evaluator and has been evaluated on a number of Java EE applications.

Mendling et al. [27] propose a meta-model integration approach to enhance the security features of Business Process Management Systems that operate using Web Services (BPEL). The meta-model elements of web services' BPEL are mapped to RBAC elements. Roles and partners in BPEL, which represent the sets of operations that are carried out during a business process, are mapped into RBAC roles. Activities, which provide a channel for an external party to send a message to a BPEL, are mapped into RBAC permissions. The authors develop an XSTL transformation script to extract an XML description of roles and permissions from a BPEL process definition which enables the definition and enforcement of RBAC polices and constraints.

There has been only a little work on UML-based security modeling [28, 29, 30, 31] . The focus of UMLsec [28] is on modeling security issues such as data confidentiality and integrity. Basin et al. propose Model Driven Security (MDS) and its tool SecureUML [30] to integrate security models into system models. The authors first specify a secure modeling language for modeling access control requirements and embed it as an extension of UML Class diagrams. The authors of authUML [29] take a step back and focus on analyzing access control requirements before proceeding to the design modeling to ensure consistent, conflict-free and complete requirements.

The Ahn and Hu method [31] differs from the above approaches in using standard UML to represent access control features of the security model. They provide policy validation based on Object constraint Language (OCL) and a role-based constraint language (RCL2000) [32], and then translate the security model to enforcement code. These efforts are forward engineering approaches, while the real need is for a reverse engineering approach that recovers and analyzes access control polices in existing applications. This is the focus of our work.

7 Conclusions and Future Work

In this paper we have presented an approach and a tool, PHP2SecureUML, to recover a role-based access control (RBAC) security model from automatically recovered structural and behavioral models of dynamic web applications. We use source transformation technology to implement the model-to-model transformation and composition. The resulting model can be used to check for RBAC security properties in the application under test. We demonstrated our approach on recovering RBAC security models for a medium-sized production web application, PhpBB 2.0. In our current work we are using the generated models to support web application security analysis, testing, maintenance and reengineering, using PhpBB 2.0 as an example, and we have recently begun a similar study of role-base security in the popular production educational support system Moodle. We are also planning to conduct a large scale evaluation to better test the effectiveness of our method, and to extend and adapt our approach to address other security analysis tasks.

References

[1] Project, O.W.A.S.: The Top Ten Most Critical Web Application Security Vulnerabilities, `https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project` (last access November 26, 2011)
[2] Pistoia, M., Flynn, R.J., Koved, L., Sreedhar, V.C.: Interprocedural Analysis for Privileged Code Placement and Tainted Variable Detection. In: Gao, X.-X. (ed.) ECOOP 2005. LNCS, vol. 3586, pp. 362–386. Springer, Heidelberg (2005)
[3] Alalfi, M., Cordy, J., Dean, T.: Modeling methods for web application verification and testing: State of the art. Softw. Test. Verif. Reliab. 19, 265–296 (2009)
[4] Cordy, J.R.: The TXL source transformation language. Science of Computer Programming 61, 190–210 (2006)
[5] Alalfi, M.H., Cordy, J.R., Dean, T.R.: Automated Reverse Engineering of UML Sequence Diagrams for Dynamic Web Applications. In: ICSTW, pp. 295–302 (2009)
[6] Alalfi, M.H., Cordy, J.R., Dean, T.R.: SQL2XMI: Reverse Engineering of UML-ER Diagrams from Relational Database Schemas. In: WCRE, pp. 187–191 (2008)
[7] Alalfi, M.H., Cordy, J.R., Dean, T.R.: Automating Coverage Metrics for Dynamic Web Applications. In: CSMR, pp. 51–60 (2010)
[8] Alalfi, M.H., Cordy, J.R., Dean, T.R.: WAFA: Fine-grained Dynamic Analysis of Web Applications. In: WSE, pp. 41–50 (2009)
[9] Sandhu, R., Coyne, E., Feinstein, H., Youman, C.: Role-based Access Control Models. IEEE Computer 29, 38–47 (1996)
[10] Basin, D.A.: Model Driven Security. In: ARES, p. 4 (2006)
[11] Paige, R., Radjenovic, A.: Towards Model Transformation with TXL. In: First Intl. Workshop on Metamodeling for MDA, pp. 163–177 (2003)
[12] Liang, H., Dingel, J.: A Practical Evaluation of Using TXL for Model Transformation. In: Gašević, D., Lämmel, R., Van Wyk, E. (eds.) SLE 2008. LNCS, vol. 5452, pp. 245–264. Springer, Heidelberg (2009)
[13] phpBB Group: PhpBB, `http://www.phpbb.com/` (last access November 27, 2011)
[14] Netcraft Ltd: web server survey (November 2011), `http://news.netcraft.com/archives/2011/01/12/january-2011-web-server-survey-4.html` (last access November 26, 2011)
[15] PHP Group: PHP usage Stats for (April 2007), `http://www.php.net/usage.php` (last access November 26, 2011)
[16] MySQL: MySQL Market Share, `http://www.mysql.com/why-mysql/marketshare/` (last access November 26, 2011)
[17] Alalfi, M., Cordy, J., Dean, T.: Automated Testing of Role-based Security Models Recovered from Dynamic Web Applications. In: WSE (2012) (submitted)

[18] Garzotto, F., Paolini, P., Schwabe, D.: HDM - A Model-Based Approach to Hypertext Application Design. ACM Trans. Inf. Syst. 11, 1–26 (1993)
[19] Schwabe, D., Rossi, G.: An object oriented approach to Web-based applications design. Theor. Pract. Object Syst. 4, 207–225 (1998)
[20] De Troyer, O., Leune, C.J.: WSDM: A User Centered Design Method for Web Sites. Computer Networks 30, 85–94 (1998)
[21] Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): a modeling language for designing Web sites. In: WWW, pp. 137–157 (2000)
[22] Hassan, A.E., Holt, R.C.: Architecture recovery of web applications. In: ICSE, pp. 349–359 (2002)
[23] Antoniol, G., Penta, M.D., Zazzara, M.: Understanding Web Applications through Dynamic Analysis. In: IWPC, pp. 120–131 (2004)
[24] Di Lucca, G.A., Di Penta, M.: Integrating Static and Dynamic Analysis to improve the Comprehension of Existing Web Applications. In: WSE, pp. 87–94 (2005)
[25] Letarte, D., Merlo, E.: Extraction of Inter-procedural Simple Role Privilege Models from PHP Code. In: WCRE, pp. 187–191 (2009)
[26] Koved, L., Pistoia, M., Kershenbaum, A.: Access rights analysis for Java. In: OOPSLA, pp. 359–372 (2002)
[27] Mendling, J., Strembeck, M., Stermsek, G., Neumann, G.: An Approach to Extract RBAC Models from BPEL4WS Processes. In: WETICE, pp. 81–86 (2004)
[28] Jackson, D.: Software Abstractions: Logic, Language, and Analysis. MIT Press, Cambridge (2006)
[29] Alghathbar, K., Wijesekera, D.: authUML: a three-phased framework to analyze access control specifications in use cases. In: FMSE, pp. 77–86 (2003)
[30] Basin, D.A., Clavel, M., Egea, M.: A decade of model-driven security. In: SACMAT, pp. 1–10 (2011)
[31] Ahn, G.J., Hu, H.: Towards realizing a formal RBAC model in real systems. In: SACMAT, pp. 215–224 (2007)
[32] Ahn, G.J., Sandhu, R.S.: Role-based authorization constraints specification. ACM Trans. Inf. Syst. Secur. 3, 207–226 (2000)