Supporting Maintenance and Evolution of Access Control Models in Web Applications

François Gauthier, Ettore Merlo
Polytechnique Montréal, Canada
{francois.gauthier, ettore.merlo}@polymtl.ca

Eleni Stroulia, David Turner
University of Alberta, Canada
{stroulia, dwt}@ualberta.ca

Abstract—This paper presents an approach to support the maintenance and evolution of Role-Based Access Control (RBAC) models with reverse-engineered SecureUML models. Starting from the Policy Decision Points (PDP) and Policy Enforcement Points (PEP) of an application, our approach statically reverse-engineers the implemented SecureUML model of an application. The SecureUML model is then stored in an RDF triplestore for easy querying and exploration. In the context of this study, we extracted the SecureUML model of the GRAND Forum, a web-based forum for the members of the GRAND (Graphics, Animation and New Media) NCE (Networks of Centers of Excellence), that is developed and maintained at the University of Alberta. Using three real use-case scenarios, we illustrate how simple queries to the extracted SecureUML can save developers significant amounts of manual work and support them in their access control related maintenance and evolution tasks.

Keywords—Role-Based Access Control, Maintenance, Evolution, SecureUML, RDF

I. INTRODUCTION

Modern web applications often enforce access control policies to protect sensitive data and operations. In the context of this paper, we present an approach to support the maintenance and evolution of the access-control policies of web applications. More specifically, our approach targets web applications that enforce access control policies through the widely adopted role-based access control (RBAC) model [1]. To give the reader an idea, content management systems (CMS) such as Wordpress, Joomla!, Moodle, Drupal or MediaWiki that are estimated to power around 27.5% of the top 10 million most popular websites [2], all implement RBAC models. GRAND Forum1, the subject of this case study, is based on MediaWiki2.

In a typical RBAC setup, at the source code level, access to sensitive resources is protected with Policy Enforcement Points (PEP). At runtime, when a user tries to access a sensitive resource, the corresponding PEP issues a request to assert whether or not the user has sufficient privileges to do so. Typically, in RBAC models, a PEP forms a request based on the identity of the requester and the privilege that is required to access the resource. This request is then sent to a Policy Decision Point (PDP), that analyzes the request and determines, in accordance with the access control policy, whether or not access should be granted. The result of the request is then returned to the PEP that grants or denies access to the resource. The PEP and PDP terms were taken from the XACML standard [3]

1See: https://forum.grand-nce.ca/index.php/Main_Page
2See: https://www.mediawiki.org/wiki/MediaWiki

The RBAC models of real-world systems evolve: as the system functionalities evolve to meet new needs of different users, so do the corresponding RBAC models. In this paper, we discuss a novel approach to supporting this important challenge. Our approach first extracts the access control policy from the PDP. It then proceeds to identify sensitive resources by statically mapping PEP to source code functions. These results are then merged in a SecureUML [4] model, that is stored in an RDF3 triple store, which developers can explore through SPARQL queries. In this paper, we explore three RBAC-evolution scenarios and we explain how the RDF SecureUML model produced by our method can be used to support developers that need to maintain and evolve RBAC models.

In the context of this case study, we applied this approach to the GRAND Forum, a MediaWiki based application that is developed and maintained at the University of Alberta for the benefit of members of the GRAND (Graphics, Animation and New Media) NCE (Networks of Centres of Excellence). In this work, we present a novel approach for statically extracting SecureUML models from web applications implementing RBAC models. We illustrate this approach with three real use-case scenarios that show how it can support developers in their access control maintenance and evolutions tasks. Finally, we document the usefulness of our approach by describing the refactorings that were undertaken in the GRAND Forum following our analysis.

II. EXTRACTING SECUREUML MODELS

SecureUML was introduced in 2002 by Lodderstedt et al. [4] as a modelling language designed to represent Role-Based Access Control models. While it was originally intended to be used in a model-driven development context to automatically generate access control models, we find that the SecureUML meta-model also lends itself to supporting maintenance and evolution of existing RBAC models.

In Fig. 1, we show the subset of the SecureUML meta-model that is relevant for this study. Features of the original meta-model that were not relevant for the investigation of the GRAND Forum were left out. The resulting meta-model captures the essence of RBAC models, as implemented in popular CMS. Users are assigned roles through RoleAssignment (RA) associations, roles are assigned privileges through PrivilegeAssignment (PA) associations and privileges protect

3See: http://www.w3.org/RDF/
functions through ProtectedObject (PO) or PossiblyProtectedObject (PPO) associations. Furthermore, roles can be organized into a hierarchy, where senior roles inherit privileges of junior roles through the Inheritance relationship.

The PPO association was not part of the original SecureUML meta-model and was added to properly represent real-world RBAC models. Indeed, in real-world web applications, it is not uncommon to encounter bypassable PEPs that cannot always prevent access to a resource. Listing 1 shows an example where the access to a sensitive resource is granted even if one of the PEPs fails. In such a case, a PPO association is created between the privileges and the resource to indicate that the PEP can be bypassed.

```java
1 if (isRole('PNI') || isRole('CNI')) {
2   // Grant access
3 }
```

Listing 1: Example of two bypassable PEPs. A user can be granted access to a sensitive resource, even if a PEP fails.

The SecureUML meta-model was designed to be overlaid on a concrete UML model. Hence, PO and PPO associations associate a privilege to a generic ModelElement. In the context of this study, PO and PPO associate privileges to functions. A PO or PPO association is created between a privilege and a function if at least one statement in the source code of the function is protected or possibly protected by the privilege.

The method to reverse-engineer PO and PPO associations from PEPs was previously published by Gauthier et al. in [5] and is based on a security-sensitive analysis that combines static interprocedural analysis and model-checking to extract the AC model structure of a system in a fast and precise way. The analysis takes into account indirectly called functions and reports possible contradictions between presence and absence of privileges in a same Control Flow Graph (CFG) path. The approach performs in linear execution time and memory complexity on the number of edges of the interprocedural CFG and has reasonable practical execution time on the investigated applications.

Inheritance relationships between roles were extracted through automated analysis of the PDP. In the context of this paper, we distinguish between explicit and implicit inheritance relationships. Explicit inheritance relationships are explicitly defined in the PDP and cause a child role to automatically inherit the privileges of its parents. In the context of GRAND Forum, explicit inheritance relationships were automatically extracted from configuration files defining the PDP.

Implicit inheritance relationships, on the other hand, result from a given configuration of the PrivilegeAssignment associations and were extracted using Formal Concept Analysis (FCA). For example, consider Table I. It represents, in the form of a binary table, a fictional configuration of the PrivilegeAssignment associations in GRAND Forum. A check mark indicates that the role on the corresponding row owns the privilege in the corresponding column. Binary tables form the input to FCA.

Starting from a binary table, FCA extracts a formal concept lattice. For example, the formal concept lattice in Fig. 2 results from FCA of Table I. Observe how FCA naturally reveals the implicit role hierarchy that was initially hidden in Table I. Starting at the bottom of the lattice and travelling up, one gains privileges. Therefore, it can be observed that PNI owns all the privileges of CNI which, in turn, owns all the privileges of both HQP and AR. Edges between roles in the lattice become implicit inheritance relationships in the SecureUML model. The interested reader can refer to Gauthier et al. [6] for other applications of FCA in the context of access control models.
TABLE II: Descriptive measurements of the SecureUML model in GRAND Forum.

<table>
<thead>
<tr>
<th># Roles</th>
<th># Privileges</th>
<th># Functions</th>
<th># PA</th>
<th># Inh. PA</th>
<th># PO</th>
<th># PPO</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>95</td>
<td>22,876</td>
<td>83</td>
<td>263</td>
<td>566</td>
<td>883</td>
</tr>
</tbody>
</table>

III. QUERYING SECUREUML MODELS WITH SPARQL

Reverse-engineered SecureUML models contain a large number of classes and associations that makes manual analysis impractical. As a consequence, automated tools are needed to query the model and verify security properties. Table II shows measurements of the SecureUML model of GRAND Forum. In the table, Inh. PA stands for Inherited PrivilegeAssignment. It indicates the number of privileges that were assigned to some role due to inheritance relationships. Other columns refer to SecureUML elements that were described in section II.

In the context of this study, we first experimented with the Object Constraint Language (OCL)\(^4\), the de-facto language for querying UML models. We found, however, that open-source UML/OCL tools failed to support SecureUML models of the size of GRAND Forum. Hence, we decided to experiment with Resource Description Framework (RDF) triplestores, a Semantic Web technology that was specifically designed to support very large graphs. Triplestores can be queried using the SPARQL language\(^7\).

Triplestores store graphs in the form of RDF triples, where a triple represents a subject-predicate-object relation. SecureUML models can be straightforwardly mapped to RDF triples where subjects are SecureUML entities, predicates are associations, and objects also are SecureUML entities. In the context of this study, the SecureUML meta-model we presented in Fig. 1 thus becomes the schema of our SecureUML triplestore. The idea of using Semantic Web technologies to store and query UML models was previously explored by Baclawski et al. in [8].

1) SELECT ?function
2) WHERE {
3) { <ROLE A> ?has ?privilege }
4) UNION
5) { <ROLE B> ?has ?privilege }
7) }
8) MINUS {
9) { <ROLE X> ?has ?privilege }
10) UNION
11) { <ROLE Y> ?has ?privilege }
12) ?privilege ?protects ?function
13) }
14) FILTER (?has IN (<secureUML:PrivilegeAssignment>,
15) <secureUML:InheritedPrivilegeAssignment>))
16) FILTER(?protects IN (<secureUML:ProtectedObject>,
17) <secureUML:PossiblyProtectedObject>))
18) }

Listing 3: Template for SPARQL queries that report the roles that can access function X

and aggregations are performed over triples instead of tables. In a SPARQL query, terms that are preceded by a ? denote a variable, terms that are enclosed in < > denote RDF literals and dots indicate joins between triples. While a complete description of the SPARQL language is beyond the scope of this paper, we refer the interested reader to [7] for more details.

Throughout our interactions with developers of GRAND Forum, we discovered that the vast majority of their queries could be summarized in the following manner:

1) List the functions that are accessible by roles A or B but not by roles X or Y
2) List the roles that can access function X

These interactions have been of an anecdotal nature, although a thorough study on developers’ interactive needs would be interesting for further research.

Listing 2 shows a template for SPARQL queries that list the functions that are accessible by roles A or B but not by role X and Y. In the template, ROLE A, ROLE B, ROLE X and ROLE Y should be replaced with the RDF literals that represent the actual roles of interest. Furthermore, there are two filters and the end of the query template. The first one allows to choose the kinds of permission assignment associations to take into account. The second one allows to choose the kind of protection associations to take into account.

Listing 3 shows a template for SPARQL queries that list the roles who can access function X. In the template, FUNCTION X should be replaced with the RDF literal that represents the actual function of interest.

IV. SUPPORTING RBAC MAINTENANCE AND EVOLUTION WITH RDF-ENCODED SECUREUML MODELS

In the context of this study, we employ three use case scenarios to illustrate how RDF-encoded SecureUML models support the maintenance and evolution of RBAC models. These scenarios represent real access control issues that developers of GRAND Forum encountered and solved in the past. In the following paragraphs, we explore these scenarios to show how different tasks that previously required manual work can now be fully or partially automated with SPARQL queries over the extracted SecureUML model. We further explain how our approach was used to identify re-factoring opportunities in the current implementation of GRAND Forum.

Scenario 1. - Individuals changing roles over time: The GRAND Forum is used by members of the GRAND NCE to input progress reports, manage projects and budgets, share

\(^4\)See: http://www.omg.org/spec/OCL/2.4/

\(^7\)See: http://www.w3.org/TR/rdf-sparql-query/
publications, etc. As people move from one project to another or as they gain more responsibilities, they might need to be assigned new roles in order to accomplish their tasks.

This first use-case scenario captures a situation where a person with the Highly Qualified Person (HQP) role had to transition to the Collaborating Network investigator (CNI) role. In the context of GRAND Forum, CNI supervise HQP and can, for example, input yearly evaluation of HQP. On the other hand, some functionalities are accessible to HQP but not to CNI. In other words, at the policy level, the CNI and HQP roles were designed to be mutually exclusive. During the transition period between her old and new position, however, the person had to have both roles at the same time to fulfill her reporting tasks. In order to prevent access violations, GRAND Forum developers manually reported every function CNI had access to, identified the problematic functions (to which CNIs should have access but HQP should not) and extended the PDP with rules that properly addressed this particular situation.

In such a situation, developers would query the SecureUML model to report the functions that are accessible to CNI but not to HQP, since those are the potentially problematic functions. Substituting ROLE A with grandforum:CNI, ROLE X with grandforum:HQP and eliminating ROLE B and ROLE Y in the template of Listing 2 yields the desired SPARQL query.

In order to list functions that are accessible to HQP but not to CNI, one simply has to invert grandforum:CNI and grandforum:HQP in the query. As a result, developers can easily identify and review HQP and CNI specific functions and grant or deny access in accordance with their security requirements.

**Scenario 2. - Individuals accumulating additional roles:** The GRAND NCE sponsors several different projects. Each of these projects has a dedicated section in the GRAND Forum and project leaders are assigned the Project Leader (PL) role. During the lifetime of a project, project leaders can vary and it is not uncommon for people to have other roles in addition to a PL role. In this particular scenario, a person who already was a Principal Network Investigator (PNI) became a project leader. The issue that arose in this context is that while PNI can evaluate PL and read the evaluations of other PNI, a person that has both the PNI and PL roles should be forbidden to: (a) evaluate herself and (b) read her evaluations from past PLs. Again, the GRAND Forum developers extended the PDP with additional rules to take this situation into account.

In this case, developers were searching for PNI specific functions that are related to evaluation. In this context, a developer would first identify PNI specific functions, using an instantiation of the template of Listing 2. She would then review the results to identify those that are related to the reading or writing of evaluations. Querying the SecureUML model of GRAND Forum for PNI specific functions returns 185 functions among the 20,000+ functions in the system, a reduction of two orders of magnitude. Starting from this reduced subset of functions, a developer would then need to filter and/or review them to identify those that are related to evaluation, a problem commonly known in the software engineering community as concept or feature location [9]. Finally, a developer would need to implement rules to prevent a PNI from evaluating herself.

**Scenario 3. - New user types imply new role specification:** The previous scenarios concerned specific individuals with particular access control requirements. In such cases, extending the PDP with a new role might be a bit excessive. However, when several different users share similar access requirements that cannot be satisfied by the existing PDP, it is usually preferable to extend the PDP with a new role instead of trying to accommodate each and every users individually. In GRAND Forum, the Associated Researcher (AR) role was created to accommodate users who needed access to a subset of CNI and PNI specific functionalities. Once the role was created in the PDP, developers had to manually investigate PEPs in order to validate the functions AR were granted access to.

Substituting ROLE A with grandforum:CNI, ROLE B with grandforum:PNI, ROLE X with grandforum:AR and eliminating ROLE Y in the template of Listing 2 yields a SPARQL query that returns the functions that are accessible by CNI or PNI but not AR. In this case, the result is expected to only contain functions that are related to the upload of reports. In cases where unwanted functions appear in the results, developers can quickly track the functions and modify the PDP/PEPs in accordance with the security policy.

**Maintenance and evolution:** In the previous paragraphs, we illustrated, through use-case scenarios, how our approach could help developers solve specific access control issues. However, reverse-engineered SecureUML models can also support developers in their day-to-day maintenance and evolution tasks.

For example, it is considered good practice to regularly perform security reviews of a system. Reverse-engineered SecureUML model can help developers in this task. By formulating queries that report the roles who can access specific security-sensitive functions (see the template of Listing 3), developers were able to identify legacy constructs that needed to be replaced or eliminated. By abstracting away details that are not relevant to access controls, SecureUML models seem to facilitate and increase the effectiveness of security reviews. More empirical data is however needed to fully validate this hypothesis.

From an evolution point of view, it should prove useful to periodically extract and compare the SecureUML models of an application. Doing so would allow developers to verify that: a) Desired modifications to either PDP or PEP were properly implemented and b) Modifications to the system did not induce undesired or accidental alterations of the SecureUML model. Given that several SecureUML models can be stored in a single RDF triplestore, comparison of different SecureUML models, over several releases of a system, can be achieved with SPARQL queries.

**Limitations:** Static analysis approximates the dynamic behaviour of an application. In the context of this study, our approach under-approximates the PO and PPO associations in the SecureUML model. In other words, the SecureUML models we extract might miss some PO and PPO relations. In a security context, we consider that it is preferable to fail to report that a function is protected when it really is (false negative) rather than reporting that a function is protected when it is not (false positive). For this study, we extended our approach with support for a few MediaWiki / GRAND Forum
specific constructs, that allowed us to reduce our false negative rate to a level that was considered satisfactory by developers of GRAND Forum. No false positive was detected.

V. RELATED WORK

The work that is most closely related to ours was presented by Alalfi et al. in [10], [11]. In [10], the authors present an approach that mainly relies on dynamic analysis to reverse-engineer SecureUML models from PHP applications. In [11], authors show how to transform the extracted SecureUML model into a Prolog-based formal model that is used to verify formal security properties. Our work mainly differs from theirs from a technical point of view (static vs. dynamic, Prolog vs. RDF/SparQL). Otherwise, we strongly believe that both approaches are complementary and that the choice of which one to use should be guided by factors like the availability of test cases for dynamic analysis, false positive/negative rates trade-offs and planned usage (maintenance vs. verification).

Our approach is also closely related to conformance testing approaches, that aim to verify that an access control policy is properly implemented in a system. Pham et al. [12] developed a static analysis to verify the conformance between policies and implementations based on Abstract Syntax Trees (AST). The static analysis presented by Naumovich et al. in [13] leverages the J2EE framework specifications to verify the conformance between access control policies and their implementations in J2EE applications. Marcus et al. [14] proposed a concept location technique that is based on Latent Semantic Indexing (LSI). Koschke et al. [15] presented an approach based on scenarios to collect dynamic data related to certain concepts. Formal concept analysis is then used to combine dynamic analysis results for different scenarios and identify the functions that are most probably related to a given concept. In their work, Poshyvanyk et al. [16] combined LSI with scenario-based probabilistic ranking to improve the effectiveness of concept location.

VI. CONCLUSION

Several of the most popular websites are powered by CMS that implement RBAC models [2]. In this paper, we presented a novel approach to statically reverse-engineer SecureUML models from Web applications and store them as RDF triples. This representation is scalable, supporting models with thousands of entities and hundreds of associations, and enables a flexible querying and exploration.

We have demonstrated the usefulness of our method with a case study with a real-world system, the GRAND Forum, a MediaWiki based application that is developed and maintained at the University of Alberta to support the administration and management of the GRAND NCE. More specifically, we showcase the usefulness of our method in supporting three types of RBAC-model evolution. In these three scenarios, we demonstrated how SPARQL queries over reverse-engineered SecureUML models reduce the amount of manual work on behalf of developers. We also explained how they help them to easily identify and replace legacy access control constructs during security reviews. Finally, contrary to other approaches that require formally defined access control policies, extensive test suites or manual inputs, our approach requires very little effort from developers to yield useful results.

As future research, we would like to observe how the SecureUML model of GRAND Forum evolves over releases and investigate how this information might be useful to developers. We hope that monitoring the evolution of the SecureUML model will help identify and correct potential vulnerabilities at a sooner stage in the development cycle. Also, the integration with dynamic analysis could be investigated in the perspective of reducing the intrinsic conservative approximation of static analysis. Special regression test suites could then be targeted at checking role / privilege changes over a system evolution.

ACKNOWLEDGMENTS

This research is funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) under the Discovery Grants Program and by the GRAND NCE.

REFERENCES