# A Verification Framework for Access Control in Dynamic Web Applications

Manar H. Alalfi James R. Cordy Thomas R. Dean
School of Computing, Queen's University, Kingston, Canada
{alalfi, cordy, dean}@cs.queensu.ca

## ABSTRACT

This paper proposes a security analysis framework for dynamic web applications. A reverse engineering process is performed over a dynamic web application to extract a role-based access control security model. A formal analysis is applied on the recovered model to check access control security properties. This framework can be used to verify that a dynamic web application conforms to access control polices specified by a security engineer.

## Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement—*reverse engineering*; J.8 [**Computer Applications**]: Internet Applications

## 1. INTRODUCTION AND MOTIVATION

Current technologies such as anti-virus software programs and network firewalls provide reasonably secure protection at the host and network levels, but not at the application level. When network and host-level entry points are comparatively secure, public interfaces of web applications become the focus of attacks [26].

In this paper, we focus on one of most serious web application vulnerabilities, broken access control. Access control, sometimes called authorization, governs how web applications grant access to functions and content to some users and not to others [1]. Depending on the access control model, sets of users can be grouped into roles, where privileges are assigned to roles rather than users. This kind of access control model facilitates the administration of user management and is called a Role-Based Access Control model (RBAC) [24].

Broken access control in web applications is considered one of the top ten web application security vulnerabilities [1]. Most web applications try to implement access control polices using obscurity, where links to pages are not presented to unauthorized users. This method of protection is not sufficient because attackers can attempt to access hidden URLs, knowing that sensitive information and functions lie behind these URLs. Attackers also try to access unauthorized objects and resources other than URL pages in an indirect way, for instance, indirect access to back-end resources such as databases.

The consequences of allowing unprotected flows to crafted requests could be very destructive, especially when the web application allows administrators to remotely manage users and contents over the web. In such cases the attackers are not only able to view unauthorized content, but also to take over site administration.

Broken access control is usually caused by an unreliable implementation of access control techniques. In many current web applications, access control polices are spread over the code, which makes the process of understanding and maintaining such rules a difficult if not impossible task [1]. To protect against this attack, access control polices should be based on a strong model that is implemented at all levels of the web application, including both the presentation level and the business level as well. Checking for authorization should be done on every attempt to access secure information, and access control mechanisms should be extensively tested to ensure that there is no way to bypass them [1].

### 1.1 State of the Art

Many methods and tools have been proposed to check for attack vulnerabilities in web applications such as SQL injection and cross site scripting [16, 17], but none of them attempts to detect broken access control attacks, either by testing or by model checking. In our previous work [7, 8], we found that many methods propose static models and tools to check static properties of web applications, and some of them try to model and check dynamic features, but none of them is able to check or even model the access control features of web applications.

In general there is little work [19, 9, 14, 3] on UML-based security modeling. The focus of UMLsec [19] is on modeling security issues other than access control, such as data confidentiality and integrity. Basin et al. propose Model Driven Security (MDS) and its tool SecureUML [14] to integrate security models into system models. The authors first specify a secure modeling language for modeling access control requirements as a generalization for RBAC, after which, they embed this language within an extension of UML Class diagrams. The authors of authUML [9] take a step back and focus on analyzing access control requirements before proceeding to the design modeling to ensure consis-


*C3S2E-09* May 19-21, Montreal [QC, CANADA], Editor: B C. DESAI

tent, conflict-free and complete requirements. The Ahn and Hu method [3] differs from the above approaches in using standard UML to represent the access control features of the security model. They provide a policy validation based on Object constraint Language (OCL) and Role-based Constraints Language 2000(RCL2000) [4], and then translate the security model to enforcement code.

All of these are forward engineering approaches, while the real need is for a reverse engineering approach that is not only able to model access control polices, but also able to check them in real applications. There is a critical need for an approach that is able to test or model check web applications to ensure that they are protected from broken access control attacks, and this is the goal of our work.

# 2. RESEARCH APPROACH

Our proposed framework (Figure 1) is aimed at recovering an RBAC security model from dynamic web applications. Based on a formal version of this model, the framework can be used to verify whether a dynamic web application conforms to the access control polices specified by a security engineer, either with a correctness check, or with a counter-example if an access control violation is encountered in the code. The framework involves two main phases:

1. Staic and dynamic reverse engineering of the web application structure and behavior.
2. Security model construction and analysis.

In the following subsections we will outline all of the framework components and the flow of data between them.

## 2.1 Web Application Reverse Engineering

In the first phase, static and dynamic analysis of the dynamic web application is used to recover the basic elements of an RBAC model [24]. We need to specify the set of users, roles, resources and their hierarchies, as well as the relations and access policies between them. Extracting static models such as class diagrams and behavioral models such as sequence diagrams help us in this regard.

### *2.1.1 Static Analysis*

The static analysis shown in Figure 1(B) extracts class diagrams that help in identifying the set of users, roles, resources and any relations between them. We have proposed and implemented [6] an automated transformation from an SQL (DDL) schema to an open XMI 2.1 UML-adapted class model. The adapted model is a tailored UML class model to represent the basic ER diagram components, including entities, attributes, relations, and primary keys. Our transformation technique is a novel one in that it is open, non-vendor specific, and targeted at the standard UML 2.1 exchange format, XMI 2.1. Although comparable commercial transformations exist, they are closed technologies targeted at formats tightly coupled to the venderŠs tools, hindering portability and preventing users from choosing their preferred tools in the development process. This analysis is supported by a dynamic analysis that may refine the class diagram, as well as recover behavioral models.

### *2.1.2 Dynamic Analysis*

Static analysis is not adequate because it does not take into account the runtime behavior of web applications. Dynamic analysis is required to perform a full security analysis, including tracking user sessions, cookies, and user inputs. To recover the implicit permissions from dynamic web applications, we have proposed and implemented an approach and tool [5] to automatically instrument dynamic web applications using source transformation technology [13], and to recover a sequence diagram from execution traces generated by the resulting instrumentation, Figure 1(A).

Using an SQL database to store generated execution traces, our approach automatically filters traces to reduce redundant information that may complicate program understanding. The elements in the sequence diagram are the interactive user and browser session, the Application Server, and the application pages and entities. The messages between these elements represent page transitions and how they affect the application entities, either with read or write operations. While our current implementation supports all versions of the PHP scripting language, the framework is not tied to any particular language and can be extended in plug-and-play fashion to other scripting languages.

Our proposed framework will address code coverage by augmenting the dynamic analysis with instrumentation for code coverage, combined with a mutation approach like that of Bellettini et al. [10] for flow coverage. This will decrease the percentage of false positives due to an analysis that results in a model that only partially covers the code (leading to verifications of properties that may in fact not hold).

Even using code and flow coverage methods, enumerating all execution paths is difficult. Ideally our framework should be able to identify all execution paths, but in some cases the human factor may be unavoidable, for instance when valid or critical information is needed to for forms, user names or passwords. Like web security scanning tools such as VeriWeb [20] and AppScan [18], we may adopt a profile-based solution which requires administrators to manually supply valid values for form fields.

## 2.2 Model Construction and Analysis

In this phase a UML-based security model is constructed based on the Basin et al. [14] security meta-model (SecureUML). A transformation from this model to a state-based formal analysis model is then performed to ease the process of security analysis and verification.

### *2.2.1 RBAC-Model Construction*

The core part of the proposed framework is the security model. In order to be able to check the web application's access-control security properties, the framework must be based on a strong security model, and be able to extract it from the source code. We construct our security model using a Role-Based Access Control (RBAC) approach, Figure 1(C). Since users are not assigned permissions directly, but rather acquire them through their role (or roles), management of individual user rights is simplified. In a role-based model, permissions for common operations such as adding a user or changing a user's department become obvious.

Our RBAC model is constructed by binding the recovered application ER model [6] with the recovered dynamic behavioral model (sequence diagram). The recovered sequence diagram is generated based on execution traces collected from the dynamic analysis part of our framework [5]. Web crawling tools that mimic user interactions with web applications, such as clicking links, filling in forms and pressing buttons [15, 25] are used to automate collecting traces,

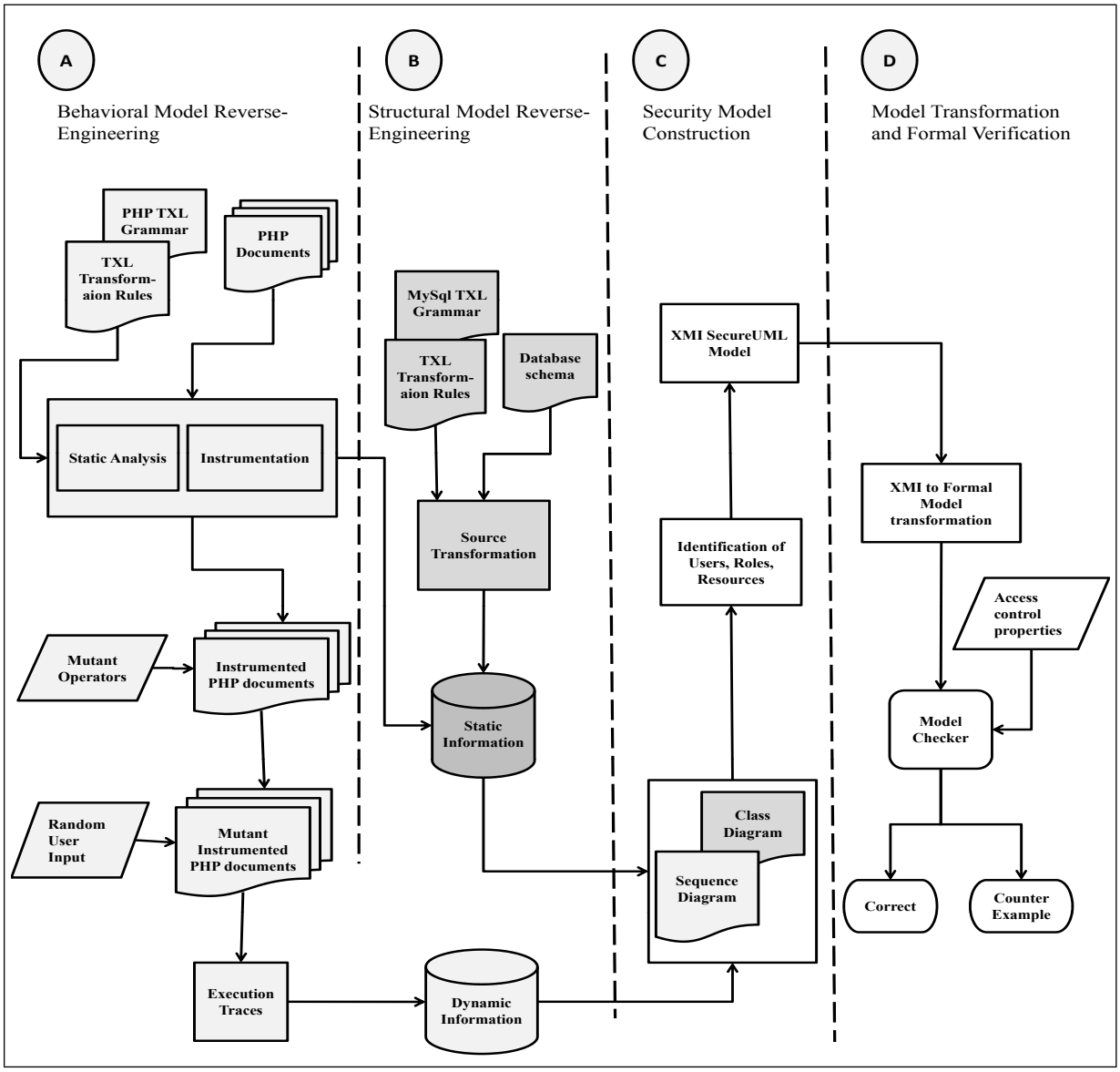


**Figure 1: The Proposed Framework**

while the application roles themselves are recovered manually by studying the software documentation. Roles can be identified from the HTTP session variable and by recovering the way the web application classifies users into roles. (Complete automation of this part is currently a work in progress). The generated sequence diagrams are combined into one single sequence diagram for the entire application in XMI 2.1 format, which is then combined with the application XMI 2.1 form of the ER model recovered by the static analysis part of our framework [6], using Model Driven Security (MDS) [14] to automatically generate a SecureUML model for the web application.

### 2.2.2 *Model Transformation and Formal Verification*

Once the SecureUML model is constructed, we need to analyze it against the security properties (Figure 1(D)). While UML models provide good support for verifying web application requirements, they need to be converted into a formal state model in order to be automatically checked [7, 8]. Several methods in the literature propose tools for the translation from UML diagrams to formal state models that can be checked using existing formal verification tools. Examples are UML2Alloy [11] and XMI2SMV [12].

We convert our SecureUML model to a formal state model using a similar conversion process. The formal model along with the desired security properties is fed to a formal verification tool such as Alloy, yielding either confirmation that the properties hold, or a counter-example. When a counter-example is generated, the problem is mapped back to the code at the function point level by tracing back to the violated dynamic page. In some cases it may be possible to go deeper, for example using the parameters provided in the URL to identify the block of code causing the violation.

## 3. EVALUATION AND PRELIMINARY RESULTS

Our approach will be validated on a number of different

web applications. Good candidate systems to assess our approach are web applications that are open source, and built using the combination of Apache server, PHP, and MySQL. The proposed framework will be applicable to other technologies as well, simply by adding their grammars to the static analysis and instrumentation stages. The most important requirement is that the web application should have some kind of permission system.

Because our approach is based on static and dynamic analysis, we require source code. Our choice of the combination of PHP, MySQL, and Apache server is based on the popularity of these technologies. According to (Netcraft)[21], Apache web server is the most deployed web server on the internet with a 58.7% market share. PHP has been the most popular server-side scripting language for years and is likely to remain so for some time. As of April 2007, there were more than 20 million websites (domain names) using PHP [22]. MySQL as well is the fastest-growing database in the industry, with more than 10 million active installations and 50,000 daily downloads [2]. The approach could be applied to other technologies as well.

In our first experiment, we are applying the proposed approach to the PhpBB [23] web application . PhpBB is the world's leading open source forum software. It has a powerful permission system and a number of other key features such as private messaging, search functions, a customizable template and language system, and support for multiple database technologies.

So far we have evaluated our prototype tools, SQL2XMI [6] and PHP2XMI [5], on PhpBB 2.0. SQL2XMI is able to automatically reverse engineer an ER class model from the PhpBB source, and PHP2XMI is able to automatically reverse engineer two kinds of sequence diagrams form PhpBB, one that represents the basic page transitions for each role, and a more detailed version that shows the effect of each page transition on the application entities recovered by SQL2XMI based on dynamic read and write operations.

# 4. CONCLUSION

The proposed approach is a novel one in web application security verification. Besides being the first approach to tackle the issue of access control verification, the proposed framework is flexible enough to allow for different server side technologies and databases in plug and play fashion.

Our approach also yields the potential for application in systems other than web applications. The static and dynamic reverse-engineering front-end of the framework can be reused for other kinds of analysis, and the framework could be used to discover other kinds of security attacks, such as cross-site scripting and SQL injection.

In our first experiment, the framework is being evaluated on one of the most popular PHP web applications, PhpBB, to check that the application is free from any remaining access control vulnerabilities.

# 5. REFERENCES


[1] The Top Ten Most Critical Web Application Security Vulnerabilities, http://www.owasp.org/documentation/topten, last access June 27, 2007.

[2] MySQL AB, MySQL Market Share http://www.mysql.com/why-mysql/marketshare/, last access Nov 26, 2008.

[3] Gail-Joon Ahn and Hongxin Hu. Towards realizing a formal RBAC model in real systems. In *SACMAT 2007, 12th ACM Symposium on Access Control Models and Technologies, Sophia Antipolis, France, June 20-22, 2007*, pages 215–224.

[4] Gail-Joon Ahn and Ravi S. Sandhu. Role-based authorization constraints specification. *ACM Trans. Inf. Syst. Secur.*, 3(4):207–226, 2000.

[5] Manar H. Alalfi, James R. Cordy, and Thomas R. Dean. "Automated Reverse Engineering of UML Sequence Diagrams for Dynamic Web Applications". In *WebTest 2009, 1st International Workshop on Web Testing, Denver,Denver, Colorado - USA April 4, 2009(in press)*.

[6] Manar H. Alalfi, James R. Cordy, and Thomas R. Dean. SQL2XMI: Reverse Engineering of UML-ER Diagrams from Relational Database Schemas. In *WCRE 2008, the 15th Working Conference on Reverse Engineering, Antwerp, Belgium, October 15-18*, pages 187–191.

[7] Manar H. Alalfi, James R. Cordy, and Thomas R. Dean. A Survey of Analysis Models and Methods in Website Verification and Testing. In *ICWE 2007, 7th International Conference on Web Engineering, Como, Italy*, pages 306–311, 2007.

[8] Manar H. Alalfi, James R. Cordy, and Thomas R. Dean. Modeling methods for web application verification and testing: State of the art. *Softw. Test., Verif. Reliab.*, 2008 (in press).

[9] Khaled Alghathbar and Duminda Wijesekera. authUML: a three-phased framework to analyze access control specifications in use cases. In *FMSE 2003, ACM workshop on Formal methods in security engineering, FMSE 2003, Washington, DC, USA, October 30*, pages 77–86.

[10] Carlo Bellettini, Alessandro Marchetto, and Andrea Trentini. WebUml: reverse engineering of web applications. In *SAC 2004, ACM Symposium on Applied Computing, Nicosia, Cyprus, March 14-17, 2004*, pages 1662–1669.

[11] Behzad Bordbar and Kyriakos Anastasakis. MDA and Analysis of Web Applications. In *TEAA(2005), Trends in Enterprise Application Architecture, VLDB Workshop, Trondheim, Norway,*, volume 3888 of *LNCS*, pages 44–55. Springer.

[12] Daniela Castelluccia, Marina Mongiello, Michele Ruta, and Rodolfo Totaro. WAVer: A Model Checking-based Tool to Verify Web Application Design. *Electr. Notes Theor. Comput. Sci.*, 157(1):61–76, 2006.

[13] James R. Cordy. The TXL source transformation language. *Sci. Comput. Program.*, 61(3):190–210, 2006.

[14] D.Basin, J.Doser, and T. Lodderstedt. Model driven security: from UML models to access control infrastructures. *ACM Trans. Softw. Eng. Methodol.*, 15(1):39–91, 01 2006.

[15] Canoo Engineering. Canoo WebTest, *http://webtest.canoo.com*.

[16] Yao-Wen Huang, Chung-Hung Tsai, Tsung-Po Lin, Shih-Kun Huang, D. T. Lee, and S. Y Kuo. A testing framework for Web application security assessment. *Computer Networks*, 48(5):739–761, 08 2005.

[17] Yao-Wen Huang, Fang Yu, Christian Hang,

Chung-Hung Tsai, Der-Tsai Lee, and Sy-Yen Kuo. Securing web application code by static analysis and runtime protection. In *the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA, May 17-20*, pages 40–52, 2004.

[18] Sanctum Inc. Web Application Security Testing, AppScan 3.5., http://www.sanctuminc.com, last access September 5, 2007.

[19] Daniel Jackson. *Software Abstractions: Logic, Language, and Analysis*. MIT Press. Cambridge, MA., March 2006.

[20] B. Michael, F. Juliana, and G. Patrice. Veriweb: automatically testing dynamic web sites. In *WWW 2002, the International World Wide Web Conferences, Honulolu*.

[21] Netcraft Ltd. November 2008 web server survey, http://news.netcraft.com/archives/2008/11/19/november_2008_web_server_survey.html, last access Nov 26, 2008.

[22] PHP Group. PHP usage Stats for April 2007, http://www.php.net/usage.php, last access June 27, 2007.

[23] phpBB Group. PhpBB, http://www.phpbb.com/, last access June 27, 2007.

[24] R. S.Sandhu, E. J.Coyne, H. L.Feinstein, and C. E.Youman. Role-based access control models. *Computer*, 29(2):38, February 1996.

[25] WatirCraft. WATIR, *http://wtr.rubyforge.org*.

[26] Adrian Wiesmann, Andrew van der Stock, and Mark. *A Guide to Building Secure Web Applications and Web Services*. Open Web Application Security Project, OWASP, 2005.